LEVEL III

REPORT No. 7901

MARCH 1979

89p.



Ross A. Bott

# A STUDY OF COMPLEX LEARNING: THEORY AND METHODOLOGIES.

Technical rept.,

N00014-76-C-0628,
ARPA Order-2284

408 267

UNIVERSITY OF CALIFORNIA, SAN DIEGO

CENTER FOR HUMAN INFORMATION PROCESSING
LA JOLLA, CALIFORNIA 92093

*This research was supported by the Advanced Research Projects Agency and the Office of Naval Research, Personnel and Training Research Programs and was monitored by ONR under Contract N00014-76-C-0628, NR 154-387, under terms of ARPA Order No. 2284. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, of the Office of Naval Research, or the United States Government.*

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>7901 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>A Study in Complex Learning: Theory and Methodologies | | 5. TYPE OF REPORT & PERIOD COVERED<br>Technical Report |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>Ross A. Bott | | 8. CONTRACT OR GRANT NUMBER(s)<br>N00014-76-C-0628 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Center for Human Information Processing<br>University of California, San Diego<br>La Jolla, CA 92093 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS<br>NR 154-387 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Personnel and Training Research Programs<br>Office of Naval Research (Code 458)<br>Arlington, VA 22217 | | 12. REPORT DATE<br>March 1979 |
| | | 13. NUMBER OF PAGES<br>163 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | | 15. SECURITY CLASS. (of this report)<br>Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Learning, complex learning, knowledge representation, instruction

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(Over)

This paper presents an analysis and model of the cognitive processes underlying complex learning situations. A theory is proposed that attempts to specify particular internal knowledge structures generated and modified during instruction, and to use them to explain specific difficulties that the learner experiences and also the overall progress being made. The theory states that (1) the underlying process reflects largely the prior knowledge structures of the student interacting with the information implied by the instruction; (2) the learning mechanisms involved are mostly simple and automatic; (3) the key information within the knowledge structures which allows complex learning to occur is the similarity between higher-order structures.

This theory was applied to novices learning to use a computer text editor by reading a basic instruction manual and completing some exercises. After each sentence of instruction, they were asked to describe their current understanding, any difficulties they were aware of, and their expectations about what would follow. Their protocols were analyzed for evidence of underlying cognitive processes. These learning processes and the associated knowledge structures on which they operate were then modeled in terms of the theory at several levels of detail. Several issues of knowledge representation related to the model are discussed and possible solutions proposed.

A Study in Complex Learning:

Theory and Methodologies

Ross A. Bott

Center for Human Information Processing
University of California, San Diego
La Jolla, California 92093

Report No. 7803
March 1979

The views and conclusions contained in this document are those of the authors and do not necessarily reflect the policy of any agency of the United States Government. Partial research support was provided by a contract from the Office of Naval Research and the Advanced Research Projects Agency, monitored by ONR under contract N0014-76-C-0628, NR 154-387. Support was also provided by National Institutes of Mental Health grant MH-15828 to the Center for Human Information Processing. R. Bott is now at Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304.

## TABLE OF CONTENTS

# I: Introduction

This paper is an attempt to describe and theorize about how people learn about complex bodies of knowledge, such as a new programming language, instructions to operate a machine, Civil War history, or the use of a computer text editor. Acquisition of knowledge domains of this type, to be called here complex learning, are characterized by the learning of not only of facts, but of a large number of interrelationships between them. In the learning situations which occur throughout life, this type of learning is much more the rule than the exception.

If one looks at this learning process from sentence to sentence during the instruction, a complicated picture emerges: The process proceeds with backtracks, sidetracks and blind alleys, even in cases where the student appears to be having little difficulty. If one is to build a complete theory of the process of complex learning, one that captures and characterizes the difficulties that student's have even in cases of successful learning, one must have a method for looking at the process from sentence to sentence. Except for a few isolated cases, such methods don't exist. For a further discussion of the feasibility and problems of such an attempt, see Appendix A.

The theory of complex learning proposed here must address several goals:





(1) Learning process: it must characterize, specify, and finally formalize what the process is that underlies the acquisition of complex knowledge domains. Such a specification can be made at several levels, from a general description to a formalization of the precise structure of knowledge within the student at each point in time, and of the processes which allow movement from one structure to the next. Three different levels will be described and analyzed here, along with a discussion of the advantages and disadvantages of each level, and especially of the issues and problems involved with moving between levels.

(2) Knowledge structure and representation: Because complex learning involves the study of complex knowledge domains, one must have a means of characterizing the structure of such domains such that their effects on learning can be analyzed in detail. Some of the issues of knowledge representation will be discussed, and a representation system proposed. In parallel to the study of learning process, knowledge representation systems can also be considered at many levels. Again, some of the issues of moving between levels will be discussed.

(3) Methodology: A set of methods must be developed for obtaining and analyzing data from complex learning situations. The basic source of data here will be protocols of students reading instruction manuals. The learning theory proposed will be used to direct an analysis of these protocols, and of the learning situation in which they occur. By using a combination of these two sources, the protocol analysis and learning situation structure, one can begin

to model the process of learning. Again, there will be different levels of methodologies corresponding to the level of theory used.

(4) Teaching: Finally, such a learning theory is *not* a prescription for teaching. However, it should eventually have something to say about how one should teach. The answer to this is difficult and equivocal, and left to the conclusion.

## A Summary and Brief Outline

The basic premise of this theory of complex learning is that all teaching information is interpreted by the student as part of his or her prior knowledge. Learning occurs as a result of the partial failure of this interpretive process, and the subsequent recovery. This theory will be progressively elaborated in the chapters to follow.

The formalization of learning processes and the discussion of knowledge representation theory in the abstract at worst can be excruciatingly boring, and at best is often needlessly difficult to follow, even to the most interested observer. In an attempt to combat this, the approach taken here will be to present much of the theory in the context of three very long examples from a particular learning situation, and similarly describe the methodologies simultaneously with their application.

Theoretical excursions and sidetracks have not been avoided completely, however. It is the nature of the study of complex learning, perhaps more than in other domains, that it is subject to the "tangled ball of string problem": Each issue unrolled and discussed tends to pull





up several more issues, and it is often difficult to decide when to just stop and cut. I haven't been entirely successful in fighting this -- hence some of the theoretical sidetracks. Those less cogent to the major points I would like to make about learning, but which I feel are too important (or dear) to omit altogether, have been relegated to the Appendices.

Chapters II through V discuss theory, experimental methods, and models in the context of the first example, a learning situation involving one command from a computer text editor. Chapter II briefly describes the experimental design, and then presents and annotates sections of protocol occurring as students learned in this situation. In Chapter III there is a preliminary formalization of what is being taught in this situation. Given this context, IV presents the general aspects of the theory of complex learning proposed here. Chapter V uses this theory to model the behavior described in II.

Chapters VI and VII discuss a more detailed example, the learning of how to type on a computer terminal. Chapter VI considers some of the issues and problems involved in representing the student's prior knowledge, which within this theory plays a crucial role in determining the learning process which occurs. A particular knowledge representation system is developed which attempts to overcome some of these problems. Chapter II applies this representation to modeling the learning behavior in this example at a more detailed level.

Chapter VIII is an application of the theory to a form of learning by analogy, and attempts to explain some examples of learning

behavior. My intention within these three learning examples is to elaborate the general theory, and the corresponding methodology, to different levels of detail. In addition, each tries to emphasize a different facet of the theory: the first on the interaction between prior knowledge and information within the instruction, the second on implicit information and determining the precise source of learning, and the third on a higher level application to learning by analogy.

Finally, Chapter IX tries to draw some overall conclusions about the theory and methods. Some tentative forays are made into the other half of the learning situation: teaching.

There is an important point to be made about the overall goal of this theory and hence of the thesis itself. In no way does the theory propose to be <u>the</u> solution to the problems of analyzing complex learning. Its goal is to carefully lay out the issues involved in developing such a theory and to propose some general mechanisms underlying the behavior in complex learning situations. Finally, by developing possible elaborations of this theory and methodology at several levels, it attempts to demonstrate the kind of analysis that is necessary to bridge the gap between educational and cognitive theory.

# II: The Print Command

All three learning examples considered in this thesis are segments of a larger instructional situation, from which all data were collected. Therefore, before beginning this first example, I will briefly describe the knowledge domain being taught within this situation, and the experimental design and methods.

## Choosing a Knowledge Domain to Teach

It might be reasonably argued that because the learning process of complex domains is so complicated, one should choose for study a domain which is reasonably simple, whose structure is known, and which is relatively unrelated to the prior knowledge of the student. However, there are strong arguments against this: The majority of complex domains which are learned in real life do not have these characteristics at all -- they are usually reasonably complicated and have a considerable degree of overlap with the learner's prior knowledge. For these reasons I have chosen to study the learning of a computer text editor. Although the students are computer novices, there is still a large intersection with prior knowledge about the process of editing and in the vocabulary used.

The computer text editor used as the domain to be taught is one called ED, and is part of the UNIX computer software system (copyright: Western Electric). It is a <u>line oriented</u> editor, that is, it operates

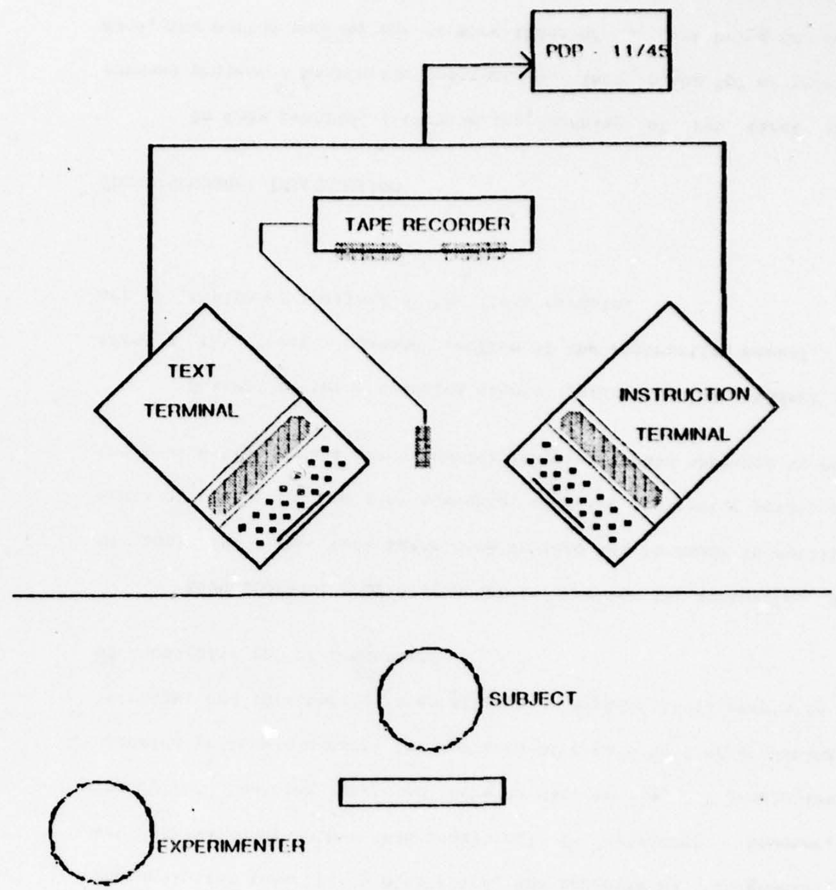
on a file of text as an ordered list of lines. The basic commands within the editor are print (a line or group of lines), delete, append (or add), and substitute (one string for another). Only the basic parts of the first three of these commands are taught within the manual used. To sketch briefly this information, "5p" would print the fifth line of a text file, "3,7p" would print the third through seventh lines, "6,8d" would delete the sixth through eighth lines, etc. The append command differs from these: "7a" causes ED to enter "append mode" starting at line 7, adding all further typed lines to the text until a line containing only a period is typed.

The teaching manual I studied was written by our research group, with the goal of teaching the knowledge domain as clearly as possible, while still attempting to keep the learner's interest. The entire text of this manual is given in Appendix B. The problems that will be found in this manual when it is analyzed in the sections to come, and the difficulties that student's have, should not be taken as a condemnation of this manual: The point of this thesis is to show how the theory and methodologies proposed here can be used to uncover such hidden flaws, and explain how and why they cause problems, even in carefully-written manuals by good teachers.

Experimental Setup

Students were placed in a soundproof experimental booth with two computer terminals (see Figure 1). The terminals were controlled by a PDP 11/45 computer (using UNIX). Each student was asked for permission to record the instruction session. They were asked to define aloud a






The Instruct System

Figure 1

few terms, and then began to read the manual.

The manual was presented one sentence at a time on the right hand terminal, preceded only by a brief set of general instructions about talking aloud (also given in Appendix B). The student could get the next sentence by pressing the space bar. The left hand terminal, initially blank except for a cursor marker, was used for the editing task, when instructed by the manual.

Students were stopped after reading each sentence of the instruction manual (and before seeing the next sentence), asked to describe what they were thinking about, and often asked additional questions (the nature of these to be discussed later). Sessions lasted one to two hours each.

Subjects

The students that were run in the experiment were all undergraduates at UCSD, participating either for class credit or pay, neither affected by their learning performance in the experiment. Subjects were asked to sign up if "they knew nothing about computers." Although most knew very little, their conceptions about what "know nothing" meant, did vary. Approximately 20 subjects were run in all.

Purpose of Data

The general goal of this thesis is to describe and explain learning behavior by theorizing about and modeling the learning process underlying this behavior. Therefore, the type of datum needed is one



which provides as much evidence as possible of this process in action during a learning situation. One of the most information-rich sources for this purpose, is protocols of a student proceding through an instruction manual, and this is the basic form of data used here.

There are problems with learning protocols, both in obtaining and analyzing them (for a discussion, see Appendix C). In general, they are observational rather than analytical in character. However, the theory of learning proposed here is descriptive, not predictive: The upcoming protocols suggest the sequence of events that occur during the learning, and this theory is an attempt to explain these events in terms of a cohesive set of processes.

Each protocol provides enough information for months (or years) of analysis. The tack taken here instead was to study in detail segments of protocols from many students, covering particular points within the instruction manual where crucial learning events appeared to occur.

Several of these learning events occur as the student reads through the "print command" section of the instruction manual. These are the topics of analysis of the first example.

The p Command: Introduction

In this example, I examine the learning of the first editing command presented within the instruction. This is the "p" or print command, the method whereby one or more lines of the text being edited can be "printed" or displayed on the terminal screen. I am interested in

how the student *learns* a basic part of this command: that typing "3p<CR>" causes the third line of the text to be printed, and in general that typing "[n]p<CR>"[1] causes the printing of the nth line. The teaching of this one rule occurs very early in the instruction manual, shortly after the introductory instructions to think aloud. The instruction sequence for the part of the command that I analyze is given in Figure 2 (for a complete listing of the instruction manual, see Appendix B).

Before reading the protocols, it is very *useful* to take the role of a teacher of computer novices, and read through the section of the manual in this figure. Consider carefully where you think learning difficulties will occur, exactly what they are, and what caused them to occur. *As noted* earlier, the intent of the writers of the manual was to avoid learning problems. And, in fact, it is often the reaction of people reading it that very few or no problems should occur, and that, if anything, the manual would be too simple for undergraduates.

Because I am studying the detailed process of this learning (and *because* the theory proposes that this process depends heavily upon a particular student's prior knowledge structure), the analysis here must be based upon a single subject, and not a group. Therefore, a complete protocol of one student proceeding through this section is presented and annotated below. However, to show that this student's behavior is not extreme or peculiar, excerpts from other student's protocols will be given. In addition, this will give an idea of the degree of variation between subjects.



THE p COMMAND

How to print text.

Type "3p" on the terminal to your left.

Type the key marked RETURN.

Every command must end with RETURN.

The 3rd line of the text has been printed.

Now try printing the fifth line.

You should have typed "5p (RETURN)"

This would have displayed the 5th line.

**Instruction Manual: Print Command Section**

**Figure 2**

<u>A Preliminary Analysis of 'Print Section' Protocols</u>

Because the instruction manual is presented one sentence at a time, the protocol is grouped accordingly. The analysis here will follow these natural boundaries: Each sentence of this part of the instruction manual will "title" the sections below. The complete protocol from one student in response to that sentence is presented, and then analyzed. After this, protocol fragments from other students are presented, as support and to indicate variations. Readers may find that they do not agree with all of these interpretations. However, the analysis and the models in later chapters will not be greatly affected by a small number of disagreements.

In the protocols below, "S" followed by a number marks a statement by a particular student, "E" a statement by the experimenter, and "MANUAL" a statement from the instruction manual. The use of multiple dots (...) indicates a pause. The student for which the complete protocol is given is S7.

======================================================================

MANUAL: THE p COMMAND

E: OK, there isn't much information there, but do you have any idea in general what will come next when you press the space bar?

*S7: ....well...the p command...*

E: Do you have any expectations...

*S7:* Well, the only think I could think of is that at the presentation of p I'm supposed to do something...or

E: What do you mean by the presentation of p?



*S7:* I would think that if I'm given a p, then I would be designated to respond to a command... that the p command would mean something that I'm suppose to do.

E: A command usually means that there is a person doing the commanding and a person being commanded. Can you speculate as to what takes the various roles in this particular case?

*S7:* I am the person being commanded, and the thing doing the commanding is the p, or the p command.

E: Any idea what the p command might be?

*S7:* No.

E: OK, go on.

The student interprets the use of the word "command" here such that she will be the one being commanded. The most immediate entity doing the commanding is "p" or the "p command." This may imply, however, that she believes that the computer or terminal is the real commander (since one of them presumably does the "presenting" in her statement "at the presentation of p").

This is an example of what I will call the <u>commander-commandee problem</u>. I will not model the source and development of this problem in detail. However, this reversal of roles within the concept of command will be modeled within the context (described later) since it plays an effect in her further interpretations. I hypothesize that she develops this idea in analogy to the instruction terminal, which asks or "commands" her to do things.

This problem is quite common among subjects (about 70%), and appears to correspond to the amount of prior knowledge about computers, and subsequent learning difficulties. In the example below, the subject sees the "computer" instead as the "commander:"

E: A command usually implies someone doing the commanding and a person being commanded. Um, who do think is the person doing the commanding here?

S20: Um, the computer.

E: And who is the person being commanded?

S20: Me, or the person at the computer.

=====================================================================

MANUAL: How to print text.

S7: (laughs) I have no idea how this relates to the p command.

E: Can you interpret that sentence?

S7: How to print text...as I told you before I like to write and so that means something else to me... What I would figure that would mean to most people would be how to run off, how to go through the action of publicating the text, but to me it means to print up a manuscript ... lettering.

E: OK.

S7: But then I was just thinking I won't be lettering anything that would be defined as a text I don't think.. I would probably be lettering just a passage... So maybe that would mean a printing, as in the mechanical printing.

E: Do you expect that that is what they will tell you to do next?

S7: I figure they are going to tell me how to organize it, how to get it done. I still don't know what they mean by a p command though.

E: OK, go ahead.

The student appears to find no satisfactory interpretation of this sentence. The difficulty appears to stem from her prior knowledge of the definitions of the words "print" and "text": Her definitions of "print" appear to include only "use non-cursive handwriting" and "use a printing press." She cannot combine either of these definitions with



"text" in any way that is reasonable within the context. The definitions she appears to lack are the usage of the word "print" in computer jargon, and the notion of "text" as a string or line of characters.

This problem was also quite *common* among students, although there *are some* differences as to how it is tackled and resolved. Two variations of this are given below:

E: What's going through your mind? When you read this, how do you interpret that? What do you think it means to "print text"?

S3: I don't know, I was trying to figure out what they meant by the sentence there, whether they are going to teach me how or what. Which I figure they are probably going to do. Not to print a text, but...

E: If somebody said "print a text" to you, what would you do?

S3: I would figure I got to write a book first (laughs).

The next subject proposes a couple possible interpretations:

E: Any interpretations?

S11: Now all of a sudden they don't have it in complete sentences *anymore*. (laughs)

E: That's true.

S11: It doesn't really follow the other thing, the p command.

E: OK, if you were to paraphrase that ...

S11: How would you print a text, or how would you ... write a text.

E: By "print a text", what do you mean?

S11: Um, ...

E: or write a text.

S11: Maybe you punch it in on the keyboard... Or ... or they want you to interpret their text, how they printed it.

Both of these subjects add the article "a" between "print" and "text" in their attempts to find a reasonable meaning (in the true meaning, an interpolated "a" does not make sense). Finally, the last subject appears to be greatly relaxing the constraints on her definition of "print" in order to find an interpretation: "to interpret their text, how they printed it."

The reasons and process underlying the difficulties with this sentence will be modeled extensively in Chapter V.

=====================================================================

MANUAL: Type "3p" on the terminal to your left.

E: Before you type 3p, do you have any expectations?

S7: That's another thing I was thinking of. That's how I thought the p was going to be interwound with it. I thought that.

E: That what?

S7: That the command p and how to organize the *text*. *See here's* a 3p, that's interwound in it somehow.

E: When you say organize a text, where do you...

S7: That's the impression I got from your going to teach me how to print the text.

E: OK, when you type the 3p, do you expect anything to happen?

S7: I hadn't thought about that.

E: So you didn't have any expectations.

S7: No, I was just going to type it.

(S7 types 3p.)

The student notes, rather vaguely, that "3p" contains a "p", and therefore is probably related to the p command. She claims to have had no expectations of something occurring when she types "3p". *I* don't believe this is entirely true: she probably expects that "3p" will appear on the screen when she types it. There is indirect evidence *for* this in her later lack of surprise at the appearance of "3p". *In addition*, at least one other subject voices this:



E: OK, when you type 3p, what do you think will happen?

S19: I'm not sure, I never ... I would assume 3p would be *on* there, but maybe ... it might bring something out. I'm not sure, or it may just print on the screen.

That the subject knows that "3p" will appear, in of itself is a non-trivial inference. Also needing explanation is why most subjects don't mention this expectation even though the event appears quite possible to them. The model presented in Chapter VII addresses these two problems.

In the first subject, I appear to plant the expectation of some event in addition to the appearance of "3p". *She retains this* expectation for the next couple sentences. This is a major slip on my part, and the information provided must be added to the model later. (On the other hand, it is conceivable that this information would have occurred to her shortly.)

=====================================================================

MANUAL: Type the key marked RETURN.

S7: Type the key marked RETURN... on which terminal?

E: On this one. (indicates the left terminal)

S7: So I'm suppose to type out "return"?

E: No there's a key marked RETURN here.

E: When you type it do you have any expectations?

S7: Yeah, I think something will happen, I think something will come out. Something will return...

At this point the student types RETURN, and the left hand terminal now looks like Figure 3.

S7: Hum.

E: Any speculation as to what's going on here?

S7: That this guy here (right terminal)... is... is telling me how to print the text... no, organize the text, and it's relating me to this one. And this (left) terminal is feeding me problems with text, such as "With complex topics problems of representation and organization".

E: Do the 3p and the "With complex..." relate in any way?

S7: No, I thought the 3p was the representation of "With complex topics..."

E: What do you mean by "the representation of"?

S7: Like I said, I thought that texts were stored in different sections of organization of knowledge, and that 3p represents... this one topic.

E: I see,... so, do you mean by represents that it's a label?

S7: Yes.

E: So what did the 3p do...I mean you typed 3p RETURN... ...

S7: I typed 3p RETURN and so if I were to say to myself what is the cosine of something is...like if I want to know what the cosine of something is that is analogous to 3p, and I'm going to go 3p and it's going to say "cosine is the tangent over sine" or something. So it's interrelated.

E: I see, OK. That's a very clear analogy. I see what you mean.

The task before the learner at this point can be characterized as follows: Three events have occurred in the situation recently -- the typing of "3p", the typing of RETURN, and the change in the contents of the terminal screen. This last event can be divided into two subevents,



3p
With complex topics, problems of representation and organization
■

State of terminal screen after typing '3p RETURN'

Figure 3

the appearance of the line of text "With complex topics, problems of representation and organization" on the second line of the screen, and the reappearance of the cursor block on the third line. The student must decide (1) how these events are related, and (2) how their interaction fits in to what has been learned up to this point.

To a student which has correctly followed the instruction manual up until this point, the relation between these events is relatively straightforward: that "p" refers to the "print command" which allows the displaying of lines of text. Therefore, 3p is probably an example of this command, where "3" is a number, perhaps a line number.

Given this description, the protocol segment above can be interpreted as the student's struggle to answer the two questions above. And, given her earlier problems, she has insufficient information on which to determine the correct relation between the three events: She has been unable to find a meaning for "print" within this situation. She appears to have little or no idea of the structure of the text file. And the roles for the use of "command" here have been reversed. She appears to struggle with the information within the events and the prior knowledge she does have: that "3p" is not a word and therefore might be a label. Also, the line of text is the beginning of a sentence and thus possibly the beginning of the text. (Her assumption that the <u>first</u> line of the text has been printed will not be clear until a couple segments later.)

This brief description glosses over many of the complexities of this crucial learning event. I will explore these and try to model them in detail later.



A similar interpretation as to the identity of the line of text which appears occurs in the protocol of the student below:

E: What do you think this is? (points to the 3rd line)

S10: (pause) I sort of think it would be the beginning of the text. It sort of doesn't say much, it's sort of a hanging part, and you think it continues.

Other student's reactions to this point in the manual ranged from no difficulty at all (about 10%) to severe problems such as these. Even among the larger problems, though, the behavior is quite variable. This is understandable: given insufficient prior information to interpret the events correctly, there is a variety of information within the events to base an interpretation on. Many of these don't become evident until the student responds to the sentence "The third line of the text has been printed". Some further examples will be given there.

==================================================================

MANUAL: Every command must be followed by a RETURN.

E: Just paraphrase that sentence.

S7: Hum... ... That they will give me -- the right terminal will give me ... the command...like 3p another command,...and I will press the commands,...in other words 3p is a command, and I will be pressing it on the left terminal, and after I press each command I will have to press RETURN again, like I have before.

E: Can you give me an example of a command here.

S7: "3p"

The subject appears to interpret this instruction sentence reasonably correctly. The process in response to this sentence will not be modeled further.

Note, though, that she solves her commander-commandee problem here. In principle, this could provide her enough information to rethink her interpretation to the last event. However, I believe it is generally the case that interpretation structures, once established, are not easily toppled except by blatant contradictory information (and sometimes not even then). Other evidence of this will be given later.

=====================================================================

MANUAL: Notice the screen.

There is no direct response to this, since she has been automatically looking at the screen to check for events. Although it is clear intuitively why this happens, the information which allows her to initiate this looking process is *quite complex and* is the subject of Chapter VII. (The information lies deep within her structure for "type" as newly modified for typing on a "terminal.") However, because of this, this statement is completely unnecessary in the manual. That it occurs is a result of the common attempt of teachers to try to keep a consistent (in this case, consistently simple) level of teaching. Yet this sometimes produces sentences like this in the same manual as "Type '3p' on the terminal to your left" and the problems which result. It is often difficult for teachers to teach from any structure but their own (an expert's structure), <u>even if they know the level of the student</u>. More about this in the conclusion.

=====================================================================



MANUAL: The 3rd line of the text has been printed.

S7: (laughs)

E: What was your first reaction here?

S7: Has been printed... ... I don't know what or where the third line of the text is. And I didn't know where on the screen it was either.

E: By "has been printed" what do you think they mean.

S7: That it's been filed back in the computer somewhere.

(long pause)

S7: Oh!

E: What?

S7: I don't know where I'm getting this impression, but, I'm starting to get this impression that 3p is the command to bring out each line of the text, and the first line of the text is "With complex...". And I haven't been presented the command for the second line either, but that there is -- there exists ... a third line, and it has been printed within the computer.

To students having little or no difficulty with this instruction manual, this sentence merely confirms their hypothesis that "3p" prints the third line. On the other extreme, students such as the one presented here have interpreted the "3p" entirely differently, and are baffled by this sentence. However, this sentence does serve to reveal these incorrect interpretations to the observer. If they have not been apparent before. Thus, her decision that "3p" elicits the first line can be inferred from her response here. In the example below, the student has concentrated on the perceptual information provided by the terminal screen. To her, the "text" must be what is on the screen, and there are only two lines there (the "3p" and the line of text):

S12: (laughs) There's only two lines over there, if that's what they're talking about. Unless they count one, two, three. (meaning the line with the cursor as the third line)

E: OK, so by that, you mean, since there are only two lines over here, what's the third line that they're talking about, and...

S12: Well, maybe it's been printed, but it hasn't shown up on the screen, yet.

E: OK, um ... ... any other interpretations you can think of?

S12: (pause, no answer)

In addition, there are some intermediate cases, where the subject was able to recover with this sentence, and appears to form the correct interpretation. Thus, the subject below had previously interpreted "3p" as: "I'd say the RETURN was what made the information to come out of the 3p slot, something like that." She responds to this sentence as follows:

E: What's going on?

S3: Well, the only thing I can think of is this is maybe part of the third line. Other than that I can't think of what else they would mean by that.

E: Of what?

S3: Of the text. This must be the third line of your text.

It is often the case that these "intermediate" students have not really settled into the correct *structure*, as becomes clear in later sentences.

======================================================================

MANUAL: Now try printing the fifth line.

S7: (laughs)

E: Why don't you paraphrase that, to start out with.

S7: I'm supposed to print the fifth line of the text. And this is a command, but it's not represented as a 3p. And... ... The only thing that I could do to print the fifth line is by relating it to the first one, and to me that is "With complex..."

E: So do you interpret the command as follows: that you should type something on the terminal that relates to this line here. Is that what you meant?



S7: Yes.

E: OK, fine.

E: Any other possibilities?

S7: ... ...No.

E: OK, go on.

S7: Well, OK, I want to say this. The only that I could print a fifth line is that if I could possibly know what the third line was. And if in the next presentation I get a third line then I could possibly print the fifth line.

E: By doing what?

S7: By doing what? By just making it up.

E: Go on.

The interpretation that the subject has established for the "3p<CR>" makes it very difficult for her to carry out this request in this sentence. In a sense, given that she believes that she should be able to carry out the request (else the instruction manual is not playing fair), there appears to be an immense number of possibilities for things to do. However, her incorrect prior knowledge about "print" acts in combination with this to narrow down the possibilities to one: the request means to type out a fifth line. Unfortunately (or fortunately), this is also not reasonable. Note that either source of knowledge alone would lead to a different interpretation.

In other problem cases, the student's inability to find a reasonable interpretation for "The third line of the text has been printed," usually leads to the same dilemma for this next sentence -- too many possibilities. Usually the salience of the prior definition of

print suggests similar unreasonable choices. For example:

S20: ...Now first they say there's three lines, and now they say the fifth line ... Maybe they want you to just type out the fifth line.

E: What do you mean by "type out the fifth line"?

S20: Well, I mean it's not five lines like 1, 2, 3, 4, 5. It's just the words "the fifth line".

This request and the prior definition of print, often reveals the instability of the correct interpretations of the intermediate students. The segment below is from the same intermediate subject whose protocol was given for the last instruction sentence.

S3: I really wouldn't know what to put because I'm missing the fourth line.

E: OK, what do you mean by that?

S3: Well, if they've given me the third line and they want me to print the fifth, I think I would need the fourth to help me out. I mean, the only think I could think of would be something to do with "representation" or "organization". (S is referring to the content of the third line)

E: What do you mean by print there? And what do you think they mean by print there?

S3: That they would want me to print something on there. (indicates the text terminal.

==========================================================================

MANUAL: You should have typed "5p (RETURN)"

S7: 5p. Do you want me to tell you what I think that means?

E: OK.

S7: I think it's going to be the third line... ..I don't know why. If they still want me to print out the fifth line, I think that would be the third line, that 5p return will...

E: OK, fine.



At this point, the student types "5p<CR>". The contents of the terminal screen now look like Figure 4.

S7: *Huh! I read that in a book somewhere.*

E: Huh.

E: Is this the third line?

S7: No, it's the second line.

E: If you were to summarize what you think is going on now...

(long pause)

S7: Well... ... I don't understand what they want me to do. Well, that the 3p and the 5p are... ... the 3p is like the mine... that the RETURN button represents my...rather the computer's recollection and organization of that knowledge, and it also at the same time prints up what that certain part of the knowledge is.

E: OK, what's the 5p.

S7: The 5p is ... the label for the second line.

E: OK, go on.

The student appears to be trying to keep her interpretation established from "3p<CR>". This leads to the hypothesis that "5p<CR>" will produce the third line (it is unclear as to exactly where this comes from).

Her conclusions after seeing the next *line* of text uncover another hidden (but totally unintentional on the teachers' part) flaw in the instruction manual: The third and fourth lines are repeated, in order to provide an editing example for the delete command later. Unfortunately, this makes the third and fifth lines seem seem to follow each other, if looked at only on a semantic basis. Given the shakiness of her hypothesis that it was going to be the third line, she "pounces"

```
3p
With complex topics, problems of representation and organization
5p
 of knowledge become crucial.  Learning is not a unitary process
```

State of terminal screen after typing '5p RETURN'

Figure 4



on this semantic fact, and concludes that the second line has been printed. (Note that if she has the correct hypothesis up to this point, the hidden "flaw" is not a flaw at all. It is very difficult for a teacher to foresee her line of reasoning and the consequences.)

It is important to point out that an integrated explanation of her behavior at this point requires an understanding and analysis of the underlying learning process: During the experiment I had no idea how she came up with the second line, and it was only in analysis of the process later did it fit together. (This comment applies to several other points as well.)

Finally, her last long comment to this instruction sentence provides evidence that, even though her structure for the p command is still wrong, at some more abstract level, it is now beginning to resemble a correct structure -- that examples of the p command display portions of the text.

The earlier student who concentrated on perceptual aspects of the screen on which to base an interpretation, has similar problems:

E: Interpretations?

S12: (laughs) Well, you didn't tell me to type 5p RETURN!

E: Any idea why they're telling you this?

S12: No, they should have told me before.

E: OK .. why should you have typed 5p RETURN.

S12: Maybe they started out: this is the third line, that's the fourth line, and that's the fifth line.

E: What's the fifth line?

S12: Where the square is.

E: OK

S12: And ... just ... guessing, just put, like say this was 4p, then just put 5p. Just like the 3p.

==========================================================================

MANUAL: This would have displayed the fifth line.

S7: This would have displayed the fifth line. This ... (laugh, and long pause)

S7: "would have displayed"...

S7: OK, maybe I tuning in on this... Maybe the "With complex..." isn't the first line, but rather the third line, and that 3 represents the number of each line, and that p represents the label of that.....and that 5 represents the fifth line, but it's represented it does not display it...this line shown is the display of the fifth line, this 5p is only the label. This is what I think they mean by "would have displayed," that 5p would have done it but it didn't actually to it, it was the recollection from the RETURN button that displayed the fifth line.

E: OK, go on.

With this statement, her knowledge structure for the p command begins to become usable. The past modal form "would have" causes some problems and makes the explanation difficult, but this is not serious and fades by her next use of the command.

The reorganization is not complete, however: In the p command as defined in the protocol, [n]p is the label for the nth line and the RETURN brings the line out: In essence, the RETURN is the "command." The "p" has not yet been separated as the label for the a class of commands. (This does not occur until later.) This final structure is quite common among the subjects in general, and in some the perception of p as a class of commands appears to occur only very late in the manual. This is quite reasonable: given prior knowledge similar to this



subject, the information provided is not sufficient to narrow down the choice of possible structures for the p command to the real structure.

Where does the reorganization which occurs in response to this statement come from? It is difficult to be precise here, but there are a few indications: The subject realizes that her current interpretation has difficulties, and this last statement may be the final straw -- to search among the possibilities which still exist and choose one which can explain all of the events better. Second, as noted before, the higher-order structure is nearly correct, and may direct the reorganization of the lower-level detailed structure.

==========================================================================

Below is a summary of the major learning events and problems occurring in the above protocols:

(1) The commander-commandee problem: Students believe that the computer is commanding them rather than the opposite. This appears to have an effect on the understanding that the p command is a subtask of editing on the computer.

(2) Problems with the definition of print: The prior knowledge of the concept of print is not the same as the definition used here. Yet the definition is close enough to be highly misleading, and this causes particular problems in the interpretation of "How to print text." This process and the difficulties caused will be modeled in detail in Chapter V.

(3) Problems with the definition of "text" and of the structure of the text file used here.

(4) The interpretation of the three events, the typing of "3p", the typing of RETURN, and the change in appearance of the screen. How do the students determine the relationship between these events, and on what information is this determination based?

(5) The problems in interpreting the sentences in the manual beginning with "The third line of the text has been printed," given an incorrect interpretation in (4).

This concludes an exploration and a preliminary analysis of learning behavior in response to the print command section of the manual. Some of this behavior is surprising, even given the tentative explanations proposed. Certainly it is the case that a superficially simple manual generated a wide range of learning behaviors and difficulties. These observations suggest a couple of further goals:

(1) This variety of behaviors needs to be explained and organized in terms of an underlying theory of learning. This theory must deal with both the specific knowledge structures involved, and of the learning processes which operate on these structures.

(2) Can this learning theory be used to generate a detailed model of the process underlying the learning of the print command?

These two goals are the subject of the next three chapters.

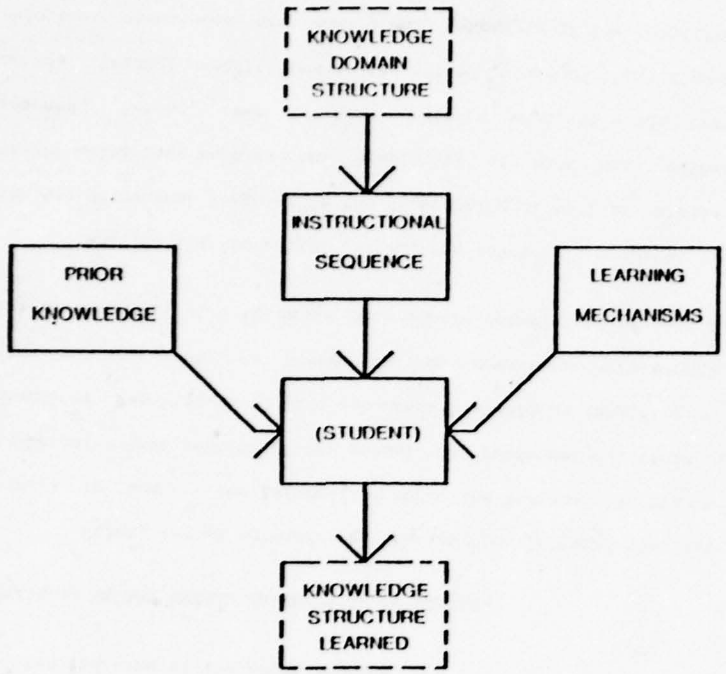
# III: An Analysis of the Print Command Knowledge Domain

Before developing a theory to explain the learning behavior in the last chapter, it is useful to consider this learning situation from a different perspective. Instead of taking the viewpoint of the student or teacher, let us take the role of an objective observer, watching the interaction between teacher and student and the passage of information from one to the other. What is the knowledge domain that the teacher is actually intending to teach? How does the information contained within the instruction compare to this? How do both of these relate to the prior knowledge of the student?

## The Instructional Situation

The learning situations that I would like to consider here involve cases in which only a one-way communication from teacher to student occurs. Thus a student reading an instruction manual is within this category, but dialog or other interactive situations are not. In addition I am studying only the cognitive factors within this situation.

Given these constraints, the instructional situation can be characterized by the following schematic (see Figure 5). The _instructional sequence_ is the information about the knowledge domain which actually is provided to the student. This structure is to be distinguished in almost all cases from a second entity, the _knowledge domain structure_ which it is intended that the student learn. For




A schematic of the instructional situation

Figure 5



example, when a teacher lectures about algebra, he or she probably presents a series of verbal descriptions interspersed with a series of examples. The structure actually intended to be taught is the structure of algebraic theory as might be described by a set of formal rules.

Internally, the student has two general types of "knowledge" to deal with these external sources of information and eventually produce "learning": a set of <u>learning mechanisms</u>, and whatever <u>prior knowledge</u> the student has that is relevant to the knowledge domain being taught.

Thus we have four possible factors to determine the process of learning. Note, however, that the student can <u>never</u> have any information from the second of these four, the knowledge domain to be taught: What is received is only whatever structure is implied by the <u>instruction</u>. To the extent that these two correspond, the student <u>may</u> learn something that resembles the knowledge domain that is intended to be learned. As will be shown in the analysis of the print command section below, the correspondence between the two is often quite low. Even when this correspondence is high, it is not necessarily the case that what the student will learn will be similar to the knowledge structure. This is dependent upon the degree of agreement between student and teacher on the meaning of the basic parts of the instruction sequence. It will be shown that a high degree of disagreement causes much of the behavior in the earlier observations. (But this is getting ahead of the story.)

Finally, it might be argued that the external knowledge domain can play an effect, in that the student can narrow the possibilities of

the structure of the knowledge domain being taught to those which have a "reasonable" structure, one which is likely given the real world. However, the rules for determining this "reasonableness" are not part of the knowledge domain being taught at all, but of the student's prior knowledge.

This chapter studies the first two of these entities, the intended knowledge domain and the instructional sequence, in the context of the teaching of the print command.

## Analysis of the Domain Intended to Be Learned

First, let us consider how the teachers *intended* the instruction manual to convey the information about the command. In Figure 6 each sentence of that section of the manual is represented, along with a plausible description of what the student should be obtaining or inferring from that sentence. Whereas this may seem reasonable enough, from the protocols it clearly is not an accurate description of what happens.

Why isn't it accurate? First, one source of problems is that the use of natural language in the *description* of what is intended to be taught helps hide much of the complexity of what must actually be learned. Second, even if one considers what is hidden behind the natural language descriptions in the second column of this figure, the resultant structures are really only *fragments* of the total framework that the student must learn in order to understand the p command.



| Sentence from instruction manual | Information intended to be learned |
|---|---|
| THE p COMMAND | This is the first editing command that will be called 'p', or possibly just use the letter 'p'. |
| How to print text. | The purpose of this command is to allow the person editing to display the text (or perhaps parts of it) on the terminal screen. |
| Type '3p' on the terminal to your left. | When '3p' is typed, it is likely to be displayed on the screen. The string '3p' is possibly the p command itself, but more probably an instance of it. |
| Type the key marked RETURN. | Request is executed. |
| Every command must end with RETURN. | Editing commands to the computer, of which 3p is an example, must be followed by a RETURN. |
| Notice the screen. | Look at the terminal screen for any developments subsequent to the appearance of '3p'. (A line of text should be noted.) |
| The third line of the text has been printed. | The line of print just noted on the screen is the third line of the text being edited. The correspondence between the 3 in '3p' and the fact that the 3rd line has been printed, should be noted here. |
| Now try printing the fifth line. | The correspondence between the third line and the 3p noted in the previous sentence should be generalized to the nth line and 'np' (at least where n is a single digit). In particular, '5p' should cause the printing of the fifth line. |
| You should have typed '5p(RETURN)' | These two sentences should provide confirming information to what has been already learned. |
| This would have displayed the fifth line. | They should appear somewhat redundant to the student and perhaps unneeded. |

Information intended to be conveyed by the instruction manual

Figure 6

The Computer-editing Environment

In order to begin to solve this problem, one needs to analyze the complete structure of the knowledge which the teacher is intending to teach. The knowledge involved with editing a text stored in a computer, can be divided into two large structures. The first, which I'll call the computer-editing environment, includes such entities such as the computer, the terminal on which the editing is done, the text to be edited, and the editing program. In addition, there are the interrelationships between each of these, e.g., how the terminal is hooked up to the computer, in what manner the text is stored within the computer, how the editing program accesses the text, etc.

A simple description is given of the objects and relationships within the computer-editing environment in Figure 7. The use of angle brackets <> here denotes a structure of entities to be specified. Some of these entities have been defined as substructures, others as sets of primitive objects, and others as functions between sets. A diagram sketching some of these relationships is given in Figure 8.

The Editing Goal-process Structure

The second half of the knowledge structure involved with editing a computer text, to be called the editing goal-process structure, is in some sense imbedded within the first half, the computer-editing environment. Given some manuscript or other piece of writing to be edited, one presumably generates a set of editing subgoals based upon the problems of the manuscript and one's knowledge about how they can be corrected. Within computer editing, the expression of each of these subgoals must



**A Model of the Computer-Editing Environment**

**ED-ENVIRON** = $\langle$ **C**, **T**, *E*, **F** $\rangle$

**C** = $\langle$ **Proc**, **Memory**, computer-input, computer-output $\rangle$

**Proc** = $\langle$ *prog*$_1$, *prog*$_2$, ..., *prog*$_p$ $\rangle$

*prog*$_i$( computer-input, $x_1$, $x_2$, ..., $x_n$ ) --> ( computer-output, $x_1'$, $x_2'$, ..., $x_n'$ )

**Memory** = $\langle$ file$_1$, file$_2$, ..., file$_m$ $\rangle$

**F** = file$_j$

*E* = *prog*$_k$

*E*( **F**, computer-input ) --> ( **F'**, computer-output )

**T** = $\langle$ *keyboard*, *display-on-screen*, typed-input, displayed-text, computer-input, computer-output $\rangle$

*keyboard*( typed-input ) --> computer-input

*display-on-screen*( typed-input ) --> displayed-text

*display-on-screen*( computer-output ) --> displayed-text

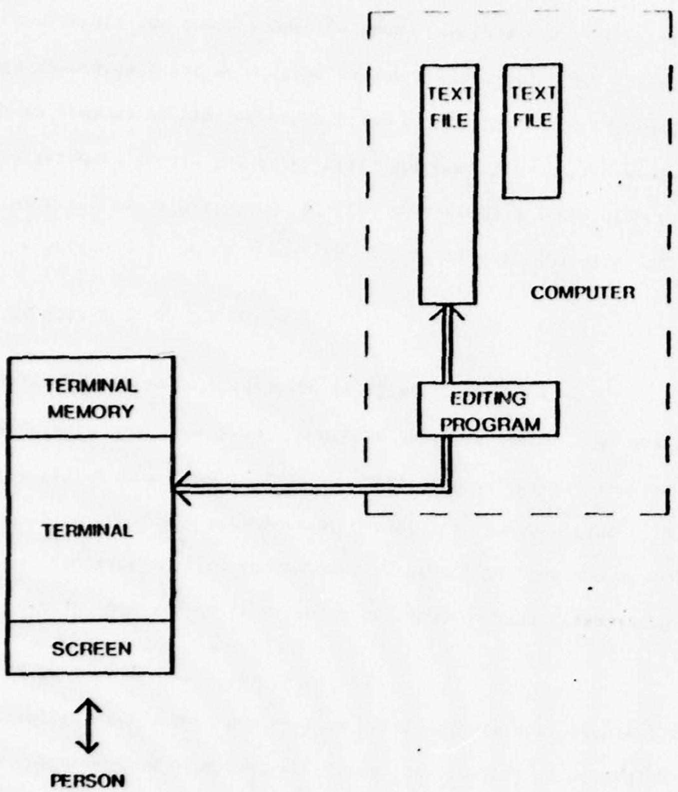
Note: In the model above,

(1) Boldface denotes a non-primitive structure consisting of other primitive and non-primitive elements.

(2) *Italicized* words are procedures, and will be described as a mapping between inputs and outputs.

(3) Words in standard type refer to primitive elements that are not procedures.

**Figure 7**




Computer-editing environment

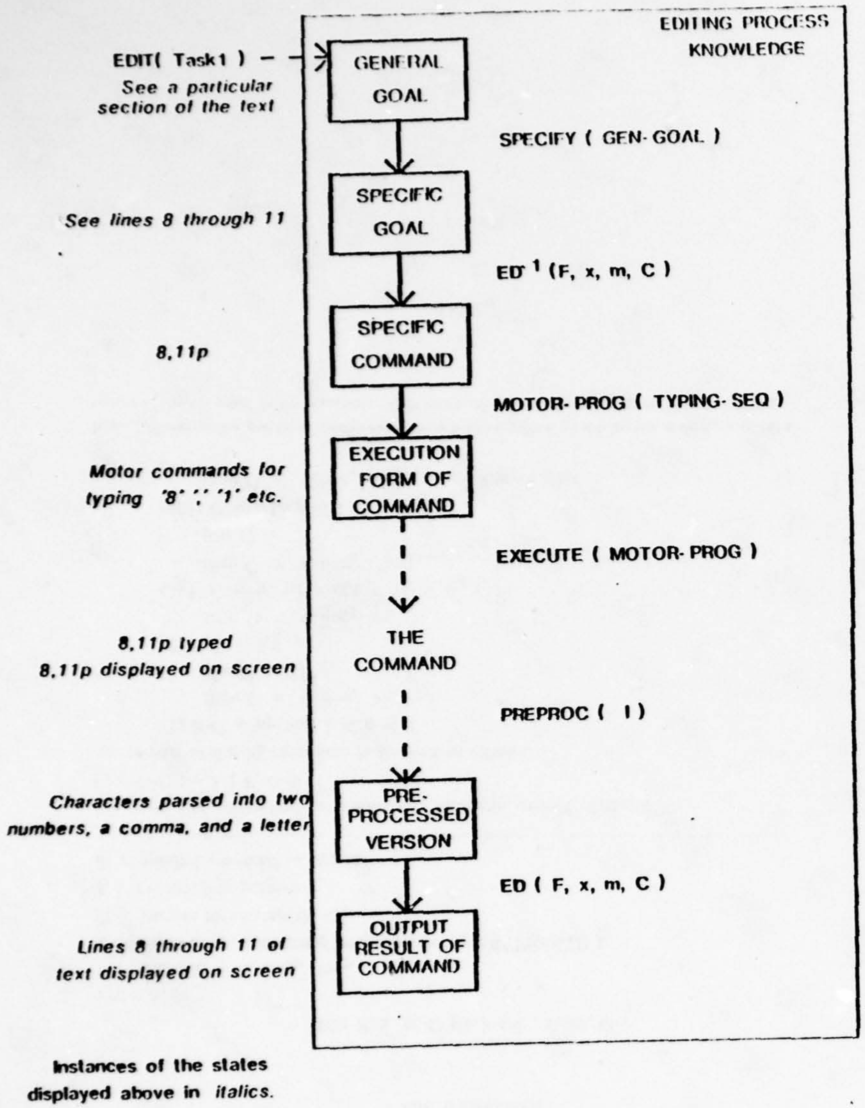
Figure 8



go through a series of transformations, involving both new goals and actions carried out, before the subgoal can be satisfied. A sketch of the structure of this knowledge is presented in Figure 9. In addition, knowledge of editing using a specific computer text editor also includes knowledge of some of the reverse transformations -- the interpretation of particular typed command strings, and what those strings cause the editor to generate. This is sketched by the last two boxes of Figure 9.

This knowledge can be viewed as a sequence of structures (the boxes), each related to the previous one by a different transformation. Once the first structure, the manuscript to be edited, has been defined, each succeeding structure is completely determined by the transformations up to that point. Thus what needs to be modeled are the structures of these transformations. The relatively small part of some of these transformations that apply to the basic use of the print command are presented in Tables 1 and 2. Note that the structures in these tables interface with the structures of the computer-editing environment described earlier. The last two transformations in Figure 9 are simply inverses of two of the earlier transformations. Two of the transforms not modeled are cognitive-motor translations; their formal structures aren't needed for our purposes here.

There are several points to note in this description about what the teachers <u>intend</u> to teach about the print command: Even sketched at this high level and just for a small part of the p command, there is an incredible amount which needs to be learned. [2] Note that much of this structure is not associated with the p command in particular but the computer editing task in general. This explains in a general fashion




Editing goal-process structure

Figure 9



## A Preprocessor of Command Strings

PREPROC( input-string ) --> command-string

Let

$a_i \in A = \{$ lower case-alphabet $\}$
$d_i \in D = \{$ digits $\}$

and the typed command string input to the terminal be:
$I = \langle c_1, c_2, \ldots, c_n \rangle$
where
$c_i \in A \cup D \cup \{,\} \cup \{\$\}$

---

Given this, PREPROC( I ) can be defined as follows:
If
$I = d_{i_1} d_{i_2} \ldots d_{i_j} a_{i_1} a_{i_2} \ldots a_{i_k}$
such that $d_{i_j} \in D$ and $a_{i_j} \in A$ and $j + k = n$
then let
$n = number( d_{i_1} d_{i_2} \ldots d_{i_j} )$
$a = alph\text{-}string( a_{i_1} a_{i_2} \ldots a_{i_k} )$
and
PREPROC( I ) = na

Note: *Number* and *alph-string* are used here as primitive procedures which convert strings of digits and characters into numbers and alphameric strings, respectively.

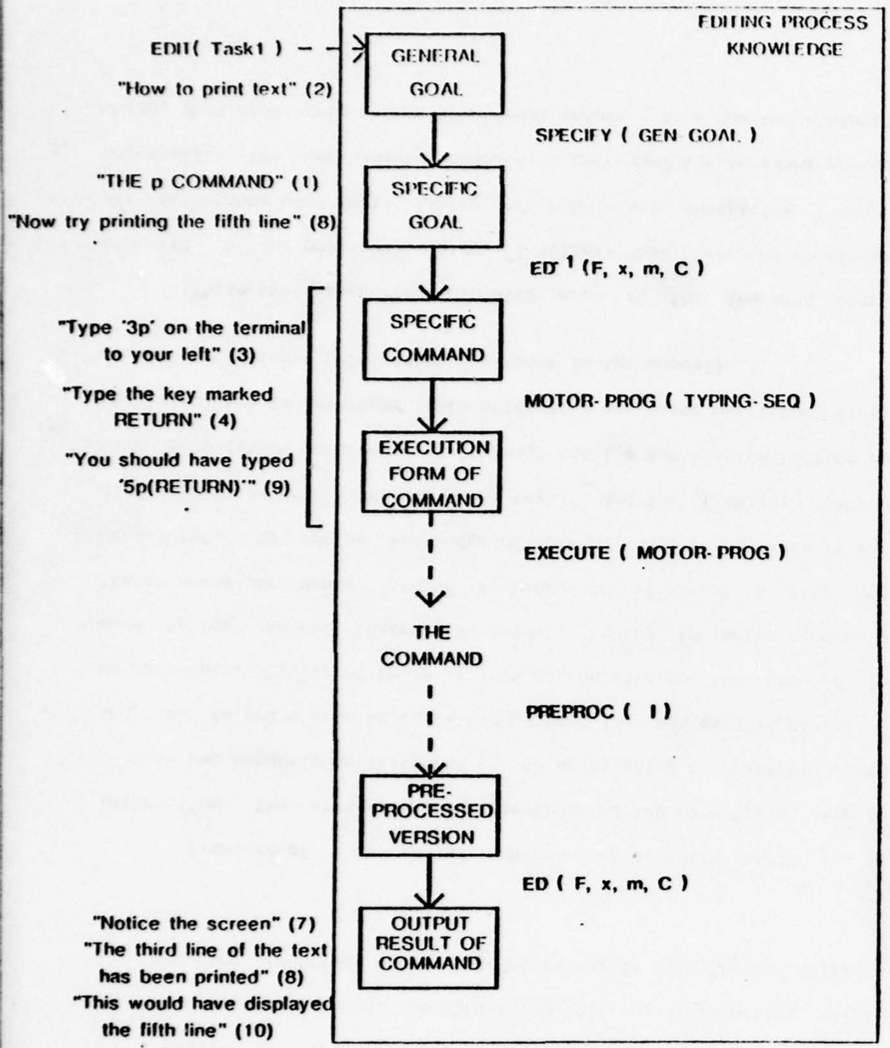
If
$I = d_{i_1} d_{i_2} \ldots d_{i_p}, d_{j_1} d_{j_2} \ldots d_{j_q} a_{i_1} a_{i_2} \ldots a_{i_k}$.
then let
$n_1 = d_{i_1} d_{i_2} \ldots d_{i_p}$
$n_2 = d_{j_1} d_{j_2} \ldots d_{j_q}$
$a = alph\text{-}string( a_{i_1} a_{i_2} \ldots a_{i_k} )$
and
PREPROC( I ) = $n_1, n_2 a$

Otherwise, PREPROC( I ) = I

---

Note: PREPROC( I ) should be viewed as a *perceptual processor* of characters, grouping them into "natural" units based upon prior knowledge.

Table 1




Instruction sentences matched to goal-process structure

Figure 10



"automatic and complex assimilation processes" to be described in the next chapter. This provides a ready-made solution to the large "hidden" gaps in the instruction manual with respect to the knowledge actually being taught: The student is assuming that these "fragments" from the manual fit into structures that he or she already knows, and therefore essentially fills in these gaps from prior knowledge.

<u>A Brief Categorization of the Observations</u>

This comparison of the knowledge domain intended to be taught and the information within the instructional sequence suggests an explanation for many of the students' difficulties observed in Chapter II. The prior knowledge of the student is insufficient to cover some of the gaps in the knowledge left by the instruction. A closer look at this explanation, however, suggests two subcases:

(1) Situations where the information provided in the instruction combined within the student's prior knowledge, is simply insufficient to determine one particular knowledge structure for the student to encode as learned.

The result of this type of situation is that there are many possible explanations for events that occur within the instruction. Examples occur in the observed interpretations of "3p<CR>", and thereafter in the interpretations of most typing events and sentences within the instruction. This phenomenon of multiple possible explanations tends to generate further occurrences, since it is highly dependent upon the current state of the student's internal knowledge: The choice of an incorrect explanation for "3p<CR>" allows many possible explanations for future

events.

(2) Situations where the student's prior knowledge <u>interacts</u> within the information provided to determine an incorrect knowledge structure to be learned. In these cases, sometimes the student will realize the incorrectness because of inconsistency with what has already been learned. In others, no contrary information exists, and the incorrect encoded structure may lead to further difficulties.

Examples of this second subcase begin quite early in the instruction: The student's prior knowledge of the word "print" appears to have two separate definitions: (1) to print using a printing press, and (2) to print meaning to letter by hand in a non-cursive style. As we have seen, neither of these is very appropriate to the use of the word "print" in this instruction manual. Second, the prior concept of "text" seems to center around a book, or at least a very long manuscript. The prior knowledge of both of these words makes it very difficult to interpret "How to print text." And the problems continue with each future occurrence of "print," until a suitable definition for this situation is developed (this often does not occur until the middle of the set of three print exercises later in the manual).

These two categories classify most of the learning events observed in the previous chapter. I believe that these two categories are reflections of a small number of consistent underlying learning processes. The next chapter develops a theory of the learning process which, when applied to the instructional sequence here and the student's



prior knowledge, explains the major aspects of these observed learning events.

## IV: A Theory of Complex Learning

The basic premises of the theory are stated below:

(1) In situations involving the learning of a complex knowledge domain, the student interprets all the information within the instruction as though it could be completely represented by knowledge structures already known to the student.

(2) If the student doesn't already know what is being taught, this interpretation process must partially fail: some of the information will not fit into prior knowledge structures.

(3) The student attempts to recover from these partial interpretation failures by finding other prior knowledge structures which can provide a reasonable meaning for the information within the instruction. The knowledge sources used in this search and choice are the student's meta-knowledge about his or her own knowledge structures, and the high-order structures of known concepts. The goal of this recovery is still to find an interpretation of the instruction within the current prior knowledge.

(4) Successful recoveries occurring as a result of (3) constitute the major part of complex learning.

(5) These interpretation processes (and hence the learning processes) are fairly simple in nature. The complexities observed





in learning behavior occur largely as a result of the complexities in the structure of prior knowledge.

All of these assumptions need to be further elaborated and formalized. This task can be divided into two subtasks -- a description of the aspects of these interpretive processes, and a specification of the knowledge structure forms on which they operate. Because of their interdependence, it is very difficult to specify one without a prior description of the other. Therefore the approach taken will be a progressive detailing, alternating between the two.

### Structure and Knowledge Representation

When talking about knowledge structures, it is necessary to distinguish between two aspects -- the form in which the knowledge is being represented, and the actual structure and content of that knowledge. This is despite the fact that the second cannot be discussed in detail outside of the context of the first.

Here the goal is to apply the theory to the analysis of a particular instructional situation, what is learned about the print command by reading through the earlier instruction manual. Therefore, I would like to emphasize knowledge structure and content here, and their effect on the learning process. Thus, I will choose as flexible a representation system as possible, and only informally specify its structure. (A formal knowledge representation system and a specification of the possible types of knowledge structures within that system is proposed in Chapter VI.)

Let us take as our knowledge representation system a simple schema or frame-like notation (I'll use the word schema here), in which each concept is represented as a schema with a collection of multiple-purpose slots. The schema itself is a distinguished "node," but otherwise has no meaning: All of the semantics of the concept being represented must be parceled out into the slots. Thus these slots will contain subparts, features, supersets, procedural definitions, other procedural attachments, etc. A sample schema for the concept of 'dog' is given in Figure 11.

There are problems with such a representation without careful elaboration, as pointed out by Woods (1975) and Brachman (1978). Some additional problems and solutions are suggested in Chapter VI. However, this system will serve for the purposes needed here.

<u>Learning Process</u>

Within the context of this representation, one can formally define the interpretive processes proposed within the first three premises above. The student's initial response to instruction is assumed to be the automatic interpretation of the information as prior knowledge. Because it somewhat resembles Piaget's concept, this first process will be called <u>automatic assimilation</u>. In terms of the representation system here it can be defined as follows: Let's assume a person has a large group of these schemas to represent all the concepts within his or her prior knowledge. Suppose a new example of an already-known concept (one that has not yet been identified as such) is input to the system. Some representation of this still-unknown information is made (perhaps in



**Dog**

- **head:**
- **body:**
- **tail:**
- **color:** (constraint - brown, red, white, black, spotted)
- **location:**
- **breed:**
- **age:**
- **name:**
- **to-find-name:** *EXAMINE-DOG-COLLAR* (procedure)
- **owner:**
- **to-find-owner:** *EXAMINE-DOG-COLLAR* (procedure)

Sample schema for the cocept "dog"

Figure 11

terms of a schema called UNKNOWN-OBJECT, with a set of slots with the various subaspects which have been identified). Then, automatic assimilation is the process by which various schemas within the prior knowledge attempt to interpret this new schema as an example of themselves.

This process has some similarities to the "viewing" of one structure as an instance of another, as used in MERLIN (Moore & Newell, 1974) and KRL (Bobrow & Winograd, 1977), among others. However, automatic assimilation is much more limited in scope (intentionally so):[4] Slots of assimilating schemas can only match like slots within the new object. Given a match, if a slot has a constraint on what can fill it (e.g., a range of values), then the value in the slot in the new object must satisfy these constraints. If all necessary slots do not find a match, then the automatic assimilation process for this schema fails. In the case that the new example is of an already-known concept, then the process will succeed for some prior knowledge schema.

Despite its limited nature, successful automatic assimilation is quite common in everyday life, where most objects and events are well-known or very usual examples of known concepts (unusual examples may begin to cause problems). The concentration of known concepts is much lower in a (complex) learning situation, and the automatic assimilation process is much less successful. The point of the assumption in this theory, though, is that process still automatically occurs, and the results of its partial failure help direct the true learning process. In addition, it generates a set of interesting problems: I will try to show that these are the major causes of the behaviors observed in





## Complex Assimilation

When automatic assimilation partially fails, the process of <u>complex assimilation</u> governs the recovery attempt. Successful completion of this second process (to be described) is proposed to provide the basis of most of the cases of true complex learning. As such it has elements of both Piaget's assimilation and accommodation.

There are two knowledge sources available to guide this recovery attempt. The first is the student's meta-knowledge, which actually includes several distinct forms: There is information about what distinguishes the various types of knowledge structures (e.g., objects, procedures, events, sets), and about how they relate (e.g., that objects play roles within procedures). There is knowledge about the probable truth and importance of known knowledge, both independently, and in relation to other concepts. Finally, there is specific procedural knowledge on how to apply these first two sources to attempting a recovery of a partially-failed automatic assimilation.

Within the representation system used here, pieces from all three forms are packaged into schemas also, <u>meta-schemas</u> that receive information about an assimilation failure, and direct the complex assimilation -- the secondary attempt to interpret the object. Thus, among this collection of meta-schemas are simple straightforward ones which suggest the relaxing of the constraints on a failing slot, or the ignoring of a non-salient feature of a new object. But they will range

to very complicated meta-schemas for dealing with more serious misfits (which, as I will try to show, are quite common in complex learning): Several of these less straightforward meta-schemas control the operation of one form of complex assimilation, to be called procedural assimilation. This is a failure recovery in which an object is interpreted by "forcing" it through a procedure in which it is likely to (but not known to) play a role. An example of this and of the possible forms of the meta-schemas involved is the basis of the model in Chapter VII.

One thing meta-schemas won't do is construct really new schemas in order to get an interpretation for new information. I believe that this type of knowledge creation is extremely rare in humans, if it exists at all. If a reasonable interpretation cannot be found by forcing it through prior knowledge, then I think that the information is usually ignored. I believe an example of this occurs in Chapter II, when student's are unable to find a satisfactory meaning for the sentence "How to print text." The underlying process as describe here will be modeled in more detail in Chapter V.

## Higher-Order Knowledge

The meta-schemas of the previous section are viewed as the "guides" of the attempt to choose a new group of prior knowledge structures which can successfully assimilate the instructional information. However, while the meta-knowledge sources are useful in this choice, the most crucial information comes from a second source, the higher-order structure of concepts.



It is easiest to describe this notion in terms of examples, and perhaps the most visible case is in procedural knowledge: The execution of many processes might be characterized by a sequence of time-ordered events, representing the execution of the subparts of the process, one after the other. However, in addition, a second representation of this same process may exist, linking only highly salient events of this process, and drawing other correspondences between the structure of these events. (Where this information comes from, and how it could be learned, is an important question, and is discussed in Chapter VII.) Given the existence of this higher-order structure, however, it can then be used for failure recovery in complex assimilation: Process which may bear very little resemblance at the lower level, may have similar higher-order structures, or at least parts may be the same, and this may be used to "relax" the lower level to allow a successful interpretation.

The overall process of complex assimilation is essentially learning by analogy: the development of a new structure based upon its similarity to a known one. The difference is that it is the higher-order structure on which the analogy learning is based. These structures provide the key information for determining which concept to base the analogy on, and for determining exactly how the concepts are similar. (In fact, in a straightforward structural sense, they may be quite dissimilar.) Because they do have the same heritage, though, I'll refer to the process underlying the complex learning studied here as directed analogy learning. Some instances of directed analogy learning are given in Chapter VIII.

This higher-order knowledge has much in common with the notions of "primitive decomposition" of verbs and other knowledge (see for example Gentner, 1975 and Schank, 1973), although the use here is somewhat different, and they follow different sets of rules. However, each concept has much to give to the other: It is clear that the study of the decomposition of verbs and nouns will greatly help the exact analysis of complex learning at a quite deep level (*for* instance, to *extend* the analysis of Chapter VII). On the other hand, the theory here suggests a process for the acquisition of such deep structures, using these same learning assumptions with a few additions (see Chapter VII). And perhaps more important, a study of the process of complex learning may provide an indirect but powerful tool for inferring these structures (by implication through the particular results of a successful assimilation).

To summarize this interpretive process, the initial response to complex knowledge instruction is an attempt at automatically assimilating the knowledge in terms of prior knowledge schemas. When this attempt runs into trouble, a second complex assimilation process attempts to find other prior knowledge schemas which allow a satisfactory interpretation. The successful choice, based primarily upon higher-order knowledge, and the subsequent successful interpretation, constitutes the learning which occurs: The new structures chosen, and the changes which are made upon them to allow interpretation, are the structures which are learned.

The next chapter attempts to apply this theory of learning process to the learning behavior observed in Chapter II, and to model it in



detail.

The crucial goal of these models is the determination of exactly what is learned. In some sense, the answer seems obvious: what is learned is the information within the instruction that is not known. However, because of the premise that all the instructional information is assumed to be already known (a restatement of assimilation), the answer is not at all clear. Thus, perhaps the key to modeling particular instances of complex learning is answering the following three questions:

(1) When and how does one decide exactly what parts of the information is really new (that is, when a partial failure has occurred)?

(2) Exactly what is the information which allows one to make that decision? Where is it located?

(3) What knowledge allows one to go about remedying the problem -- recovering from the partial failure? How does this process occur?

(4) Finally, how do the knowledge structures used in recovery relate to exactly what is learned?

## V: Modeling Learning Process in the Print Command

### A Summary

Below is a summary of the major learning events and problems observed in the protocols of Chapter II:

(1) The commander-commandee problem: Students believe that the computer is commanding them rather than the opposite. This appears to have an effect on the understanding that the p command is a subtask of editing on the computer.

(2) Problems with the definition of print: The prior knowledge of the definition of print is not the same as the concept used within the manual. Yet the definition is close enough to be highly misleading. This causes particular problems in the interpretation of "How to print text," but very similar problems occur in every later use of the word "print" within this section of the manual.

(3) Problems with the definition of "text" and of the structure of the text file used.

(4) The interpretation of the three events, the typing of "3p", the typing of RETURN, and the change in appearance of the screen. How do the students determine the relationship between these events, and on what information is this determination based?

(5) The problems in interpreting the sentences in the manual from "The third line of the text has been printed," given an incorrect





interpretation in (4).

(6) Finally, a reorganization of this interpretation into a structure which begins to resemble a correct understanding of the print command.

*In this chapter I will* analyze three of these phenomena in more detail. I will try to show how the phenomena above are manifestations of learning events within this underlying process. A common thread through the three examples is that the learning process (by assimilation) is very much the same. The variety in the learning behavior observer, results from this process operating on different prior knowledge structures. A more formal model of one of these learning events will be presented at the end of the chapter.

The information provided by this instruction manual is of two *distinct forms: verbal information* given by the sentences themselves, and visual, motor, and temporal information within the editing examples used. As can be seen from the analysis in Chapter III, these sources taken together still leave large gaps in the information the student must actually learn in order to correctly understand the print command. The information to fill these gaps must be supplied by the student's prior knowledge. The interpretive process described in Chapter IV provides an automatic mechanism for supplying this knowledge. Despite the differences in the verbal and non-verbal information forms given by the manual, these process which occur are very similar. However, because of its automatic nature, problems occur: exactly what knowledge is needed is not always as clear to the student as the teacher intended it to be.

Filling Roles in 'command'

Consider how a student can respond to the phrase "THE p COMMAND." Within the context of computers the usage of command usually implies that the operator is commanding the computer (e.g., many statements within programs are commands). If the situation concerns editing on computers, then command forms are needed to carry out certain editing tasks. A student who has this prior knowledge can interpret this phrase correctly.

If one considers this information in terms of schemas, one can imagine a "COMPUTER-COMMAND" schema, with the correct roles for commander and commandee already filled in, automatically assimilating the word "command" here. In turn, this schema would be assimilated by a "COMPUTER-EDIT-TASK" schema, since COMPUTER-COMMAND could be used to fill in a 'subtask' slot of this latter schema.

If the student does not have this prior knowledge, though, other prior knowledge may assimilate this occurrence of the word "command": Students have been reading a manual which has requested them to do several things already, e.g. "read the sentences aloud" and "say what they think will happen next." In addition, they have been told that they will have to do tasks "on the terminal to [their] left." To sketch the result of this, students in interpreting these earlier sentences will have created an instance of a COMMAND schema, in which they take the role of the commandee, and the terminal (or computer: many students see them as synonomous) is the commander. In the absence of other prior knowledge, this COMMAND schema can assimilate the occurrence of the word "command" here. Hence the commander-commandee problem occurs.



The Meanings of 'print' and 'text'

A similar problem occurs in the automatic assimilation of the next sentence, "How to print text," but the visible result of the process on learning behavior is different. The phrase "How to" suggests the upcoming explanation of a task likely to be unknown to the student. (This last statement actually includes several pieces of knowledge and inference: semantic knowledge about "how to," knowledge that the current context involves a learning situation, and the inference that in a reasonable learning situation, the teacher intends to teach knowledge which should be unknown to the student. It is likely that all students here have essentially this prior knowledge.

However, given this, most students are unable to find a reasonable interpretation of the rest of the sentence, using their prior knowledge of "print" and "text." If "print" is interpreted as "write non-cursively," then "print text" would mean "write a book," given the usual meaning of "text." (Note that this interpretation actually requires the addition of the article "a" between "print" and "text." And as noted, some subjects did in fact insert this article in their attempts at understanding.) There are at least two problems with this interpretation in addition to the missing article: Students already know how to write non-cursively, violating the current context. Also, printing a whole book would appear to be an unlikely task.

If print is interpreted as "use a printing press," then the meaning of "text" as a book makes more sense, since one usually

"publishes books." However, again the article "a" is really needed, and aside from this, it is difficult to assimilate a printing press into the current context.

I believe that the result of this combination of prior knowledge causes both automatic assimilation above to fail, and the recovery attempt by complex assimilation to fail also. No interpretation structure can be built for this sentence, and it is essentially ignored. A more detailed model of this assimilation process is given at the end of this chapter.

The consequences of the non-interpretation of this sentence extend through the rest of this section, for its purpose was to establish an overall goal for the p command. Students who fall into this difficulty must interpret the subsequent events in this section as isolated, rather than as part of a developing structure.

The knowledge which is needed to interpret this sentence correctly, is the definition of print used with computers: to print is "to get the computer to display in some form part of its memory." The meaning of "text" used above is also wrong, but I believe that this is not so much a case of a missing definition, as of a rare use of the word (i.e., text simply as a string of words). Lacking other information, the prototype meaning of "text" was used. Thus, once a proper definition for print is found by the student, a non-prototypical use of "text" is not a difficult step.

For many subjects, the development of the right definition for print occurs slowly throughout this section of the manual. The



underlying process is one of complex assimilation -- the attempt to adjust or relax current structures for print until a structure is found that works. Shreds of evidence for this process appear to occur within the observed behavior: the trying out of definitions of "organize," "type," "interpret," and "print within the computer." This process is hampered by an important difference in the higher-order structures of the definitions: the degree to which the subject of print participates in the act of printing -- very high in print(write), much less in print(printing press), and almost not at all in print(computers). Since the most important source for guiding the complex assimilation is the similarities between higher-order structures, this difference, especially between the first and last definitions, makes it very difficult for the process to proceed.

<u>Explaining Sequential Events</u>

When the student follows the requests "Type '3p' on the terminal to your left" and "Type the key marked RETURN," a series of events occur:

(1) the typing of 3p
(2) the change in the contents of the terminal screen from a small white block alone, to a "3p" printed to the left of a small white block
(3) the typing of RETURN
(4) the change in screen contents from (2) to a "3p" above a printed line "With complex topics, problems of representation and organization" above a small white block

(I have actually omitted many other events, such as the motor feedbacks from each of the typing movements, the noises involved, the movements of the experimenter during this time, etc.) The reason for the occurrence of each of these events must be accounted for, as well as an explanation

for why the events occurred consecutively in the order that they did. The same interpretive process operates here, but, because of the state of prior knowledge doing the assimilation, the visible effects of this process are different.

The explanation of some of the events and event pairs are intuitively clear: The event of typing "3p" occurs because the student intentionally executes a particular motor program for typing the characters "3" and "p." The motor feedbacks and keyboard noises which occur simultaneously with the typing event can be explained because they are expected as part of the student's knowledge of typing. In terms of the interpretive process and schemas, the particular example of the TYPE schema for typing "3p" has slots associated with motor feedback and noises, and automatically assimilates these events when they occur. Note that both the <u>occurrence</u> *of the feedbacks and noise, and their* <u>relationship</u> in time to the typing event (i.e. simultaneity) are both being accounted for in the same assimilation.

In a similar fashion, the immediate appearance of the "3p" on the screen following the typing event, can be assimilated. The process here is somewhat more complicated, though, for the position of the white block has also changed. In addition, the subjects described in Chapter II all had not typed on a terminal before, and even the appearance of the "3p" involves considerable complex assimilation. (A detailed modeling of the process underlying this one explanation is given in Chapter VII.)



The occurrence of some of these events and the relationship between them are not so easily assimilated by the subject with the prior knowledge at hand. This is particularly true of the line of text, and of an explanation of why it occurs almost immediately after the typing of RETURN. At this point, let's step back and consider the process from a different viewpoint.

Theoretically, each event within a series can be related in many ways to some or all of the other events (or not at all), depending upon the underlying structure of which they are part. The information associated with each of the events suggests clues. For example, in the case of the three events, the typing of "3p", the typing of RETURN, and the appearance of the line of text, one can note that "3" is a number, that "p" has occurred before, as is the first letter of "print", that RETURN caused an immediate change in the screen contents, that the syntax of the line which appears implies the beginning of a sentence. Not all of this information may be relevant.

The learner must create from this information a "chain of explanations" relating these three events, perhaps hypothesizing the occurrence of other unnoticed events as part of the chain. Even given the constraints of the information, there may be several choices of explanation, or possibility chains, from among which the learner must decide which is the correct one.

In order to interpret these events correctly as described, the learner must have acquired sufficient knowledge from the instruction (or had sufficient prior knowledge) to narrow the number of possibility

chains down to one. And that chain of explanations must be the knowledge which is intended to be taught.

The interpretive process can be viewed within this context: At some points in the sequence, the occurrence of and relationship between the events is easily assimilated -- the choice of on possibility chain is clear. At other points the prior knowledge is insufficient to provide a single interpretation -- the possibility chain branches into several paths. The decision as to how to solve this probably related to the learning strategies of particular students: whether to decide absolutely upon one interpretation, to take one chain tentatively while being aware of the others, or perhaps to break down at the prospect of having to choose from the array of possibility chains.

The behaviors observed in Chapter II in response to the appearance of the line of text can be explained as the visible effects of this process. Of the subjects who had difficulty at this point, all had had problems in interpreting the previous two sentences of the manual. Thus, they had less structure with which to decide among the possibility chains available. Each had to choose from among the possibly relevant clues to decide upon an explanation. Most subjects utilized the information that the appearance of the line almost immediately follows the typing of RETURN, and interpreted from this that the two events were causally related. One subject (see p. 24) used information from the syntax of the line, and inferred that it was the beginning of a sentence, and that this was relevant to the explanation. This, combined with the semantic information of the line that it could be the beginning of a paper, allowed the decision that the line is the first line of the



text. The decision was made that the "3p" was related to the text line. However, the information that "3" is a particular number was chosen as irrelevant to the explanation; instead, it is decided that "3p" as a whole was simply a label.

A second subject (see p. 24) utilized mostly perceptual clues. Other subjects (not given) searched for an explanation from among the semantic meaning of the line of text. The nature of the interpretive process is the same in each case; just the choice of information is different.

Towards Formalizing the Interpretive Process

The second of the three learning events described above, the non-interpretation of the sentence "How to print text" will be modeled somewhat more formally below. Considerable description was needed in the examples above just to explain very small segments of the learning situation, and even these skipped over some of the relevant factors. The model below is even more difficult to describe (and the same probably holds true for reading it). Therefore, it is useful to consider what purpose it serves.

My ultimate goal is to analyze the learning process to the stage that it can be implemented on a computer. If this can be done, benefits of the analysis can be realized in working models of the student. This can be used both to study further the learning behavior, and because a model of the student would be a valuable addition to both computer and human teachers.

The approach to this goal taken here is a progressive elaboration of the learning process, gradually breaking away from a dependence upon verbal description and the vagueness it allows. This approach has strong similarities to that of Charniak's in his painting program (1977). The model below is an intermediate step, still partially dependent upon verbal description, and leaving some areas vague. A more formal step is given in Chapter VII.

One problem is that each step covers a progressively smaller segment of the learning. Because of the complexity of the process, this is unavoidable at this point. The goal of this thesis in part is to explore the feasibility of this approach.

The verbal descriptions above explicitly or implicitly refer to a set of additional issues which have an indirect effect on the learning process which occurs. Before beginning an explicit model, these must be dealt with:

Context. The learning processes which operate here share with most understanding processes the use of (and need for) a "context." However, the usage of the concept "context" is slightly different here, in both creation and operation. Here contexts are generated automatically, as the learning processes proceed through an instruction manual: A group of schemas collect that have participated in or supported various successful or partially successful automatic and complex assimilations. Those from successful processes will have all their slots completed. Unsuccessful automatic assimilation will tend to generate incomplete schemas, ones that are setting off complex assimilations, or waiting for



further input. Those from completely failing assimilations will tend not to be added to the context.

As a group, this context of schemas functions by limiting the range of future processes: Essentially they direct new interpretations by automatically assimilating any new information, just as any single schema would. The result is a maintaining of the current hypothesis by forcing an organization on any new information which will support it. When something cannot be assimilated in this fashion, meta-schemas must be called in and a complex assimilation occurs.

Several examples of this process occur within the print section protocol during the "reign" of the hypothesis that 3p represented the first line of the text. The eventual complex assimilation appears to occur in the last statement analyzed, "The would have displayed the fifth line."

Teaching Postulates. These are directly analogous to "conversational postulates" (Grice,1975). There are some interesting differences, however, in the content of the rules which apply. Because learners assume that they are being taught new information, they will often reject plausible interpretations of a statement on the grounds that they already know it. For similar reasons, their tolerance for references to unknown concepts and events tends to be higher. Finally, one of the more subtle rules is that students expect a consistent level of detail within the information to be taught. Information which violates this level in either direction will often be misinterpreted.

Time. Clearly the representation of time will play a crucial role in representing most knowledge domains and the learning of them. This is especially so in learning situations such as the one here, where some events (i.e., the typing of commands and the computer's responses) occur in real time, and where the order and precise spacing is important.

The representation of time used here is as follows: Subjectively, time is perceived as an ordered sequence of time "instants" or intervals, which are timeless within themselves. I am assuming that intervals are in the neighborhood of one tenth to five tenths of a second long, with the length varying with the number of events occurring within the interval. Events within the same interval are perceived as simultaneous. This is certainly not a sufficient representation for psychological time, but it is satisfactory for our purposes here.

Perhaps the easiest way to define the role of this representation of time is to describe its use in a closely related concept, causality. If events are perceived as occurring in adjacent (subjective) time intervals, their perceived adjacency is always recorded. Unless there is known information to otherwise explain the adjacency, such a pair of events is assumed to be causally related. The noting of a causal relation between two events sets off a complex assimilation process to satisfactorily explain this causality.

### A Model of the Interpretation of 'How to print text'

The knowledge representation system used for modeling here is essentially a free-form extrapolation from KRL (Bobrow & Winograd, 1977). I believe that there are some general problems with frame-slot



notations such as this. However, they have the advantage shared by few other systems that knowledge described in the notation is easily read and intuitively fairly clear. I will also skip over many of the technical aspects of KRL; the goal here is to sketch the process and the knowledge which is needed to carry it out.

In character, this representation system will resemble the schema system described earlier. The various aspects of a concept are represented by slots. The "slot name" gives the aspect referred to, and the "contents" of the slot is the value of that aspect for the particular concept. Thus slots can include features, subparts, roles or participants (e.g. of a procedure), arguments (e.g. of a relation), and operations (to be explained shortly).

In addition, each schema has associated with it a set of processes. The *process of automatic assimilation* described earlier is implemented within this model as an automatic attempt of a schema to fill its slots with values, using input information (in this case, instruction). The knowledge as to what provides a suitable value within a particular slot is contained within the "constraints" and "prototypes" (described later) associated with that slot. The automatic assimilation processes of each schema are assumed to be capable of operating in parallel.

All schemas within prior knowledge are assumed to be constantly "aware" of new information input to the system. When an aware schema finds new information which satisfies one of its slots, it "creates a new copy" or "instance" of itself. This new instance is now "active" in

that it attempts to fill the rest of its slots, by assimilating other new input information. When it is unable to do so, assimilation fails, and complex assimilation must take over. But a model of this won't be necessary until Chapter VII.

Schemas can become active through other processes: An already active schema can instantiate and activate other schemas, during its attempt to fill its slots. In addition, schemas can be activated if it has a "trigger" slot, which activates its parent scnema when a particular condition occurs in the current context or external world. (In some sense, all slots are "triggers" for schemas, as defined here.)

Finally, schemas can have "operations" associated with them -- processes which can instantiate or fill slots of other schemas, or even "reject" or deactivate currently active schemas. While the assimilation processes above are <u>implicitly</u> part of the schema structure, these operations are explicitly stored in slots of the schema. An operation is to be distinguished from a "procedure" within a slot: The first is part of the process of the <u>knowledge system</u>; the second is a representation of a piece of prior knowledge.

A few other issues and notational conventions need to be described. These are given below and are followed by the model itself.

<u>Constraints</u>, <u>prototypes</u>, <u>defaults</u>. One issue is how to represent a person's knowledge about what is possible and what should reasonably fill particular slots within a schema. Three forms of this knowledge are used here: (i) constraints, or limits on the character of knowledge to fill the slot which cannot be violated, (ii) prototypes, which concern the usualness of the knowledge, or how well it fits the slot given that it fits at all, and (iii) defaults, to be used when information to fill the slot is not explicitly given. If no default is given, the prototype is used.



<u>Types and tokens</u>. A second point is the so-called "type-token" distinction between generic and specific knowledge. (The issues and problems underlying this distinction are discussed in detail in the next Example.) Here, token schemas will simply be prefixed by an asterisk and a number (since there can be more than one), while generic schemas have no prefix. Simplistically, the generic schemas taken as a group can be interpreted as the prior knowledge structure of the student. To save space, however, not all generic schemas will be presented beforehand.

Within the constraints and prototypes, concepts referred to (such as pencil or newspaper within the PRINT schemas below) are not modeled explicitly here. But if a word is in all capital letters, it is being modeled. The symbol ">" used here can be interpreted loosely as "the concept which precedes is limited to the constraints which follow." Finally, comments are placed in double parentheses.

```
Process of assimilation of the statement "How to print text"
=================================================================

((First, the current context at the point right before this sentence
is processed must be modeled. This will consist mostly of schemas
activated by earlier sentences within the manual.))

*1-GLOBAL-CONTEXT:
        goal: *1-LEARN
        situation: *-experiment
        environment: *-booth
((These last two tokens will not be explicitly modeled.))
---------------------------------------------------------------------------
*1-LEARN (relation)
((Note that LEARN is not a procedure as represented here. The
learning occurs as a natural product of the assimilation process.))
        teacher: *1-INSTRUCTION-MANUAL
((The student probably believes the instruction manual is a vehicle
for some unknown teacher to teach by, but this isn't important here.))
        learner: *-SELF
        knowledge-to-learn: *1-p-COMMAND
        gen-subject-area: *1-COMPUTER
                          *2-COMPUTER
                          *1-EDIT
((Since the subject does not know what "computer editing" is
yet, these are included as relevant concepts but not yet related.
The two instances of COMPUTER reflect her unclear notions as
to its exact identity.))
---------------------------------------------------------------------------
*TEACHING-POSTULATE-ONE (procedure)
        trigger:
                the current GLOBAL-CONTEXT has goal= a LEARN
        learn-schema:
((Once the TEACHING-POSTULATE-ONE is activated by the trigger,
it examines all LEARN examples which come into the current
context, to check for violation of the rule contained here.
I'll assume that it has already checked *1-LEARN, which doesn't
```

```
violate the rule.  The problem will occur when *2-LEARN
comes into existence.))
        knowledge-to-learn:
((Obtained from the slot within the LEARN instance.))
        operation: If the knowledge-to-learn is an instantiated
prior schema ((i.e., the knowledge is already known)), reject
the current LEARN assimilation.
--------------------------------------------------------------------------
*1-INSTRUCTION-MANUAL (object)
        sentence1: "We are going to teach you how to edit a text"
        sentence2: "The text is stored in a computer"
        sentence3: "There are a number of mistakes in the text"
        sentence4: "We would like you to correct them"
        sentence5:
((at this point, unfilled))
        sentence6:
        sentence7:
--------------------------------------------------------------------------
*1-COMPUTER (object)
        identity: *1-OBJECT
((That is, the PDP-11/45 the subject passed by in coming to the
experimental booth.))
        description: *-large-metal-square-machine
                constraints= a machine>
                                logical
                                does not make mistakes
                                very fast
                                processes massive amounts of information
                                electrical
((The constraints used to help fill this slot are given because
the generic schema itself is not.))
        location: *-in-room-outside-booth
--------------------------------------------------------------------------
*2-COMPUTER (object)
        identity: *2-OBJECT
((That is, the terminal on which the subject is carrying out the
editing examples.  (This schema is given next.)
The fact that this probably implies that there
is then a second computer, the other terminal, does not greatly
affect the process here and will not be modeled.))
        description:
                constraints= a machine>
                                logical
                                does not make mistakes
                                very fast
                                processes massive amounts of information
                                electrical
        location: *-in-booth-in-front-of-SELF
--------------------------------------------------------------------------
*2-OBJECT (object):
        (identity is anchored to the external world by visual information)
        description:
                constraints= none
```



```
        name: "terminal"
        part: *1-KEYBOARD
        part: *1-SCREEN
--------------------------------------------------------------------------
*1-EDIT (procedure)
        object-to-edit: *1-TEXT
                constraint= written or printed prose
((It is assumed here that the procedure for editing films is
a whole separate concept with different constraints and prototypes.))
                prototype= a manuscript from a book or paper
know which of the collection is to be used.))
                = procedure-collection:
((There appear to be at least two types of procedures.  In one,
to be called "linear-sequence-proc", the subprocedure are
executed in a specific order, with each beginning after the
previous ends.  In the other, "procedure-collection", the
order of execution is not specified, and subprocedures can
be executed repeatedly or not at all.  At a more micro-level,
of course, things become more complicated.))
        operation:
((At this point this slot is unfilled; the subject doesn't
know which subprocedure will be referred to or used.  In a
sense, the procedures below are constraints on what can fill
this slot.))
                        CORRECT-WRITTEN-ERRORS
                        DELETE-TEXT
                        ADD-TEXT
                        PICK-OUT-IMPORTANT-PARTS
                        REORGANIZE-TEXT

CORRECT-WRITTEN-ERRORS (procedure)
        object-to-be-corrected:
                constraints= written or printed prose
                prototype= a manuscript
        operation:
                procedure-collection:
                        CORRECT-SPELLING-ERROR
                        CORRECT-GRAMMAR-ERROR
                        CORRECT-PUNCTUATION-ERROR
                        CORRECT-USAGE-ERROR

CORRECT-SPELLING-ERROR (procedure)
        trigger: FOUND( spelling error )
        operation:
                linear-sequence-proc:
                        CROSS-OUT( spelling-error )
                        WRITE( correct-spelling )
((These two subprocedures of EDIT have been modeled in order
to sketch an idea of the complete structure of EDIT.  The
exact form of the rest of the structure is unnecessary for
our purposes here.))

((This concludes the model of the current context prior to
```

```
reading the sentence.))
=========================================================================

((The reading of "How to print text" now activates the following
schemas:))

*2-LEARN
        triggered-by: "How to"
((The interface between semantic parsing and the activation of
schemas is not modeled here.  Intuitively, though, the phrase
"how to" in a sentence like this, usually signals that information
will be given about the operation of what follows.))
        teacher: *1-INSTRUCTION-MANUAL
        learner: *SELF
((Both of these are gotten from the *1-LEARN in the current context.))
        knowledge-to-learn:
((This will presumably be filled by what follows in the sentence.))
        gen-subject-area: *1-COMPUTER
                          *2-COMPUTER
                          *1-EDIT
((Again gotten from *1-LEARN.))
-------------------------------------------------------------------------
*1-PRINT-WRITE (procedure)
        agent: *SELF
((Implied by the syntactic and semantic structure of the sentence.))
                prototype= human
                default= *-SELF
        instrument:
                constraints= object>
                                capable of making marks
                prototype= pencil or pen
                default= pencil
        surface:
                constraints= object>
                                (reasonably) flat
                                can be marked on
                prototype= paper
                default= paper>

        material-to-print:
                constraints= string of letters
                prototype= short passage of English prose
                default= 1 or 2 sentences of English prose
((This default is probably dependent upon the context, i.e.,
what is a reasonable task demand.))
        style:
                constraints= non-cursive
                default= all upper case letters
((This probably varies across people.))
        operation:
                = PWRITE(agent, instrument, surface, text, style)
((This is a primitive motor procedure for executing handwriting.))
        result:
```



```
                = APPEAR( the text, on the surface )
-------------------------------------------------------------------------
*1-PRINT-PRESS (procedure):
        agent: *-self
                constraint= human
                default= a printing press operator
        instrument:
                constraint= machine>
                                capable of TYPE( letters )
                default= a printing press
        surface:
                constraint= paper
                prototype= white newspaper
((This prototype would tend to vary across people.))
        material-to-print:
                constraints= prose>
                                long length
                prototype= English prose>
                                book or newspaper
                default= newspaper
        operation:
                = RUNOFF$( agent, instrument, surface, text )
((Again a primitive operation.  The details of its working are
probably vague to most people.))
        result:
                = APPEAR( the text, on the surface, in typed form )
-------------------------------------------------------------------------
TEXT (object):
        definition:
                constraint= a passage of English prose
                prototype= a book
                default= a textbook
        text-name:
                constraint= a title
        location:

=========================================================================
```

The processes which occur at this point are a combination of (1) attempts of the newly activated schemas to fill their slots (by automatically assimilating available information), and (2) attempts by the active schemas within the context to assimilate these new schemas. While I believe that many of these processes occur in parallel, for description they can be divided into several process chains:

*1-PRINT-WRITE attempts to fill its slots: "agent" has already been filled. "instrument" and "surface" are not mentioned by "How to print text," and so they are filled by default by the new instances *-pencil and *-paper. (On a desert island these would not be so easily filled, but within this context the student could assume that they can be reasonably supplied.) Information to fill the slot "material-to-write" is supplied by the sentence. The definition of TEXT does not at all fit the prototype or default information of this slot, but it does

satisfy the constraints, and *1-TEXT is used to fill this slot. (In some people, the constraints of this slot may be such that they can't be satisfied by TEXT; in this case, the process would fail at this point.) Finally, for "style" the default can be used to fill the slot. So *1-PRINT-WRITE may successfully complete its automatic assimilation.

The slots of *1-TEXT can be partially filled. The subject has been told in the manual before this section that "The text is stored in the computer." Therefore, "location" can be filled with *1-COMPUTER or *2-COMPUTER, depending upon which candidate the student believes is the more likely computer. However, no title for the text is given, and there is no default for the "text-name" slot. It is possible that this information will be given in the future, and thus this schema will actively attempt to assimilate new input information to fill this slot. (In fact, some subjects decided that "p" was the title of the text.)

Given that *1-PRINT-WRITE has successfully completed its assimilation, *2-LEARN can complete the *filling* of its slots: The "knowledge-to-learn" slot is filled with *1-PRINT-WRITE.

However, at this point, *1-TEACHING-POSTULATE-ONE within the current context can fill its slots: "learn-schema" is filled by *2-LEARN, and "knowledge-to-learn" is filled with *1-PRINT-WRITE (this is actually retrieved from the slot within *2-LEARN). Its operation is then triggered, which causes the rejection of *2-LEARN and *1-PRINT-WRITE, essentially rejecting this line of interpretation.

The second chain of processes, revolving around *1-PRINT-PRESS, is much shorter. The definition of TEXT fits the prototype of the "material-to-print" slot here much better that in **1-PRINT-WRITE*. However, here the "instrument" slot is not as easily filled: the default of a printing press is highly unlikely to be supplied given the current context. Thus, the automatic assimilation here fails at this point.

## VI: On Representing Knowledge

The next two chapters are an attempt to push the formalization of this learning theory to a somewhat deeper level. In order to do so, several of the issues underlying the development of knowledge representation systems must be considered. Some of these have been introduced, but not discussed in the previous chapters:

(1) The informal model at the end of the last example helped to reveal the huge gap between a verbal description of a learning process and a formal model. There are many issues to consider, some of which were glossed over: What is "activation" of schemas? How and why does "instantiation" occur? Is there a type-token distinction? Some of these will be considered here, and an attempt made to organize the answers in terms of an underlying process.

(2) Knowledge representation systems like the schema-slot notation used in the last chapter hide other issues. A couple of the problems will be discussed and possible solutions suggested.

(3) A second type of problem in most cognitive representation systems is that they begin with a corpus of prior knowledge frames (as I did in the last example). Where do they come from? How is the frame structure itself acquired? A set of underlying learning postulates are proposed here. These attempt to satisfy two criteria: the learning processes should resemble those which operate on the

frames after they have been created (i.e. there should be an underlying continuity in learning processes). Second, the processes should be psychologically reasonable.

(4) Finally, although higher-order structure has been implicated as the key information for effecting the actual complex learning, neither the process nor the structure have been formally described. An example of this type of structure and its operation is given here.

It may not seem necessary to consider all of these issues in a paper on complex learning. But the actual <u>source</u> of the learning is quite an elusive object, and it can only be finally pinned down at this level.

The approach taken in this chapter will be a consideration of the theoretical issues first, then a formalization of a representation system. In the next chapter, an example will be presented of the learning process in terms of this system.

<u>Developing a Basis for a Representational System</u>

It is often the case that when one chooses a particular knowledge representation system (to be called a KRS) in which to model a knowledge domain, the structure of that KRS affects how one goes about doing that representation, and in the end may bias what actually gets represented. Therefore, it is useful to consider exactly what a KRS is. What are the constraints on its structure? Given these, how does one go about building a KRS?



The process of using a representational system to describe the knowledge within some domain has *many* similarities to the notion of building a <u>formal model</u>: In a KRS there is a knowledge domain to be represented, and one intends to build, using the representational system, some structure which corresponds to this domain (or part of it). In the case of a formal model, there is an "external world" and a model which one is building whose structure proposes to correspond to part of the world's structure. By "correspond" in both cases, presumably we mean that some or all of the properties of the domain or external world are "preserved."

It is therefore useful to present a formal definition of a model, and then to consider the relationship between a representational system and this construct.

I will take as given some "world" to be modeled, consisting of a set of entities (e.g., but not necessarily, objects) E and a set of relations on these entities:

$$W = \langle E, r_1, r_2, \ldots r_n \rangle \text{ where } r_i: E^j \dashrightarrow E^j$$

In developing a model one wishes to define a new set of entities E' and a new set of relations on E':

$$M = \langle E', r'_1, r'_2, \ldots r'_n \rangle \text{ where } r'_i: E'^j \dashrightarrow E'^j$$

such that there is a mapping f from M to W is at least a homomorphism, that is, for each $r_i$ and $r'_i$:

$$f[r'_i(e'^j)] = r_i(e^j) \text{ for each } e'^j \text{ of } E'^j \text{ and } e^j \text{ of } E^j$$

If in addition E' and E have the same number of elements, then f is an isomorphism, and we can call M a <u>complete</u> model.

We can use this same formalism for models as a notation for the process of representing knowledge. However, there are some important differences. First, the correspondence here is <u>not</u> between a model and a KRS, but between a model and the <u>result</u> of applying the KRS to a particular knowledge domain -- in a sense, the result of the application is a model.

What then is a knowledge representation system? To say that it is simply a <u>notation</u> in which to construct a model, is informally correct but incomplete. The set of entities and relations within a model must be *represented* by some set of symbols in order to present and discuss the model. A representation system usually implies that one is going to construct a group of models from it. The KRS chosen constrains the nature of these models by limiting the <u>total</u> set of entities and relations from which to sample. In this sense it *is more* than a notation; it is perhaps better viewed as a partially specified model of some "world", from which true models can be drawn. (In analogy to the "theory spaces" of Appendix A, a particular KRS defines a <u>model space</u>.)

Formally, given a "world" as defined above, then a <u>KRS for this world</u> can be defined as the following structure:

$$K = \langle E^*, r'_1, r'_2, \ldots, r'_K \rangle$$

where $E^*$ is a set of entities and the $r'_i$ are relations, and K is subject to the condition that for some subset E of $E^*$ and some subset of the relations of *K*,

$$M = \langle E, r_{M_1}, r_{M_2}, \ldots, r_{M_n} \rangle$$

is a model of W above. (Note that the number of relations within K must be at least the number within W but can be considerably *larger*, and that many *of them* can have no correspondence at all with those in W.)

To describe a *knowledge representation* system in these terms seems to imply that they have very little inherent structure, and consequently little effect on the models within the associated model space. However, it does imply one important constraint: the set of possible entities and relations with which to build models has been completely determined: A <u>*primitive* level</u> for the models within the space has been specified.

The reason for introducing the concept of a primitive level is that, in order for a model from a representation system to be "meaningful," it is crucial that the primitive level within the representation system be well-specified, and <u>consistently maintained</u>. Some current representation systems, notably *many of those* based upon semantic networks, don't faithfully follow this requirement, and hence there are problems in interpreting what they represent. (Most commonly this occurs in the primitive definitions of the links between nodes -- the ri's in the notation above.)

This is basically the same point, but in a somewhat different context that Woods (1975) and Brachman (1978) make. The solution, however is not <u>necessarily</u> a small (but powerful) set of well-specified primitive relations (as, for example, Brachman uses): The choice of a primitive level is a separate issue which must be considered in the context of several other issues:

(1) the use of the representation system,

(2) what psychological assumptions are to be made, and

(3) modeling or implementation considerations.

These will be discussed within the next two sections.

One presumably develops a KRS in order to construct representations (or "knowledge models") of particular knowledge domains. What needs to be examined further is the use of these knowledge models. It is worthwhile to continue the comparison to "normal" models: In psychology especially, the attempt is to choose a mathematical structure to use as a model of some set of phenomena such that

(1) all the data found so far can be explained in terms of it, and

(2) one can derive implications from the structure that predict other phenomena which can then be tested (see Palmer, in press).

However, one can extend this -- in some sense there are actually two sources from which one can get these implications: One can derive some using the normal mathematical laws of inference. Others derive when the model used is one actually occurring in nature (or nearly so). An extreme example of such a "nature" model is modeling three-



dimensional scenes using photographs. The point of this distinction is that in the second case, the process of inference uses, in addition to normal laws of logic, the internal processes of the modeler (and of the viewer of the model). However, <u>because the goal of these models is to obtain testable predictions</u>, either source of inference is acceptable.

In contrast, models of knowledge domains can have a variety of uses, usually related to the general goal of modeling behavior in a "non-trivial" way. These can be classified into the general categories of systems which answer queries, and systems which "learn," or somehow modify themselves in response to new information (a system of the second type usually demonstrates its learning by answering queries or otherwise producing output).

In either case, the requirements on the structure chosen for the have a very different emphasis:

(1) One must be able to retrieve information about the knowledge that was explicitly modeled. This is analogous to the first model type. However, there the attempt is to organize a set of phenomena into a coherent mathematical structure, in order to obtain predictions. Here it is often sufficient to have the knowledge just enough organized to permit retrieval.

(2) One may or may not be able to get from the representation information about knowledge that has not been explicitly encoded.

If one can, this information must come from some form of logical, or semi-logical, inference from the structure of the knowledge already encoded. It <u>cannot</u> occur through inferences drawn from a

representation which depends upon the external structure of nature. In analogy to a nature model, these will be named "nature representations." The reason for this, is that, in order to draw implications from such a nature representation, the process of inference requires a model of that external structure. This second model is actually the real one, but it is hidden within the inference process.

This pitfall is not as easily avoided as might be seemed: In many versions of schema or frame-like theories, the names of the slots have exactly this problem: The semantic structure of the word used as the slot-name is implicitly used as part of the concept represented by the schema. One must even be careful of the graph-like nature of the unit -- that the schema "has" or is represented by a group of ordered slots again implicitly adds information.

Up until now I have tried to avoid the use of the process-pair cliche, but this last point can be usefully rephrased in these terms: It is possible to hide within the process implicit gaps within the representation, either by making use of hidden information, or handling inherent ambiguities. If the interpreter is allowed to do this, it also becomes very difficult or impossible to compare two models of the same domain.

## A Checklist for Development

The issues just described suggest a set of key factors to consider in constructing a KRS:



(1) What is the primitive level?

(2) What is its relation to the process which operates on the system?

(3) redundancy and implicit knowledge

(4) learning of the structures

(5) psychological issues

(6) modeling and implementation

Each is described further below:

A useful method for avoiding the last problem described in the previous section is to be constantly aware of and specify completely a primitive level for any model constructed from that KRS. If this is done, it will simultaneously completely specify the interpretive process for any model constructed: To the extent that entities at the primitive level are <u>atomic</u> or have no internal structure, then the interpretive process is a simple search and output mechanism. To the extent that the entities are not atomic, and their internal structures have an effect, then this information must be contained within the interpreter, and it must be stated what and how this is done. The trap, which more than occasionally catches someone, is when this is not done. Specific examples will be covered when the KRS used here is described.

It should be noted that the usual inference processes used are different for the normal and knowledge models. In the normal case, information, or predictions, are obtained by <u>inference of hidden redundancies</u> within the knowledge structure, redundant in the sense that it provides no new bits of information. In the case of knowledge

models, many inferences are extrapolations from perceived redundancies within the knowledge structure. For example, to reason by analogy is to notice a partial similarity (perceived redundancy) between two knowledge structures, and to assume that the rest of the two structures are similar. A second instance is generalization from a set of examples of a concept, to form a prototype or generic form.

It has been said (essentially) that the real intelligence of knowledge models is in the ability of the interpretive process to notice, and act upon such possible redundancies. In order for this to occur, the primitive level must be such that it does not hide such redundancy, either in terms of implicit knowledge, or as internal structure of primitive atomic elements. This is a second key consideration when deciding upon the best primitive level, and will be used extensively in the construction of models from the KRS used here.

A third consideration is whether the KRS models are to *learn* from new input information. Intuitively, this would seem to argue for a simple set of primitive entities, which could be more easily handled by a learning mechanism, and combined in to form new structures. However, this is actually crucially dependent upon the set of assumed learning processes; in those proposed in the theory here, the opposite is probably true.

This introduces a fourth consideration in developing a KRS and specifying a primitive level. In psychology, the choice of both a learning mechanism and what types of structures one should allow, should be at least partially determined by psychological issues. My bias here



is towards as simple of a learning mechanism as possible, and that knowledge structures within the model are such that they are constructible from these learning mechanisms. (While this may seem to be a truism, the majority of current representation systems do not consider how they might be learned, and in fact this would pose quite a problem for some of them.) The assumptions used here will be described formally in a later section.

The final considerations are issues involved in the modeling and implementation process. The major goal of this model and thesis is to analyze and describe the learning of complex knowledge, with its representation being an intermediate but necessary step in the process. Therefore, the choice of primitive level will be adjusted according to what is needed to describe this behavior. This will result in a somewhat variable level of detail. However, the procedure for evening this out is straightforward, and in the case of any important interactions this is followed through. Secondly, it can be argued that parts of the structures proposed here would be grossly inefficient if implemented on a (serial) computer. I believe, however, that efficiency considerations should be dealt with after the fundamental problems have been worked out. In the worst case, implementation will just have to wait for a bigger computer.

## Formal Description of the Knowledge Representation System

As a first approximation, the knowledge within any given domain might be characterized as a group of interrelated concepts. And, at

least on a superficial level, these concepts as a whole are not a homogeneous group: some clearly represent objects, others refer to relations between objects, features of objects, states of the world, changes in states, or procedures (just a partial listing). Any knowledge representation system chosen must be capable of representing all of these. In addition, one may want to show how some of these superficially different entities may reflect the same underlying principles, and build these into the KRS. This is the approach chosen in this model, where here the principles are assumptions about the basic psychological processes about the input, acquisition, and modification of this knowledge. The assumptions below are organized in the following manner: Premises about the structure of the system are given first, including a formal description of the KRS to be used here. This is followed by the set of processes which are used to acquire this structure, and those which operate on it.

Structural Assumptions

Knowledge domains are assumed to be divisible into a group of concentrations of knowledge, each referring to a particular "concept," where a concept may be an object, a feature, a relation, a procedure, or one of the other categories to be named later. For historical reasons, these concentrations will be called schemas. Although they will have some similarities to the schemas of Bobrow & Norman (1975) and Rumelhart & Ortony (1977) (as well as to those in other frame-like theories), there will be important differences.

These schemas will be highly interconnected by mechanisms to be described later. It may appear that allowing this makes it very



difficult to distinguish a "knowledge package" theory such as this from "flatter" representation systems such as semantic network theories. There is a crucial difference, however: in a schema-like theory, one can access in a straightforward manner all the information directly related to a particular concept. In a flatter theory, where there are no "boundaries" between concepts, one can access related information by crossing links; however, one cannot tell when the information is no longer related. (If boundary "markers" or some other mechanism is put in, then it becomes a schema-like theory.)

The KRS used here consists of four sets of distinct structures, each with a common structure type, but different from the other three. These correspond approximately to schemas referring to objects, relations, "classes" (described later), and procedures:

$KRS \dashrightarrow \{ O^*, R^*, C^*, P^* \}$

$O^* \dashrightarrow \{ O_1, O_2, \ldots, O_o \}$

$R^* \dashrightarrow \{ R_1, R_2, \ldots, R_r \}$

$C^* \dashrightarrow \{ C_1, C_2, \ldots, C_c \}$

$P^* \dashrightarrow \{ P_1, P_2, \ldots, P_p \}$

Object schemas. Each schema member of the object set is either a primitive object schema, or is non-primitive and has a group of other object schemas as its "subparts". As a result of the process assumptions below, a non-primitive object schema is actually simply the representation of the simultaneous occurrence of its subschemas. The basic equivalence of this to the notion of subparts is a result of the structure of the external world (of which the object referred to is a

member) and the perceptual mechanisms, such that the subparts of an object in the external world are usually perceived in the same subjective time interval. The concept of simultaneous occurrence will be noted in this KRS by square brackets.

The set of "object" schemas includes both schemas which refer to real objects in the external world, and to those which refer to features of those objects. This has advantages and disadvantages: The perception of features of an object will occur in the same subjection time interval as the perception of the object's subparts; this information is represented automatically by the notation here.

However, there clearly are differences between objects and features of objects: In fact, in most systems they are represented entirely differently. Here the awareness of the distinction between subparts and features is contained entirely within the parts and features themselves -- to anthropomorphize, an object schema can only find out whether each of its subschemas are object or features by having that subschema tell it. (If such information were needed, it would have to be one of the processes that each object schema were capable of. However, it is not needed here.) There are undoubtedly subtle problems with such a simplistic approach, but it goes surprisingly far, and is sufficient for our purposes here.

The set of primitive object schemas corresponds to the set of entities E* in the earlier notation for a general KRS. The non-primitive object schemas, and the members of all three of the other sets would be the relations $r'_i$ of that structure.



$O_i \dashrightarrow O_i[\ O_{j_1},\ O_{j_2},\ \dots,\ O_{j_i}\ ]$

$O_i \dashrightarrow O_k\$$

("\$" after any schema will be used to mark it as primitive)

Finally, it is important to note that the subschemas of an object schema are an unordered collection. I have intentionally left out the concept of slots and associated slot names common to most frame-like knowledge representation systems because, as argued in the previous section, part of the knowledge being represented is then hidden within the slot structure and slot names.

Relation schemas. These schemas, representing relational information, are elements of the second set above, and include a couple of intuitively quite different types of knowledge. Perceived spatial relations between objects in the external world are represented straightforwardly by schema elements such as ABOVE, NEAR and WITHIN. These spatial relations need not necessarily be static; thus PASSING-IN-FRONT-OF, and simply MOVING (with respect to a background) are also relation schemas here. However, changes in the "state" of an object, and appearances and disappearances are also represented here: All of these are characterized by two entities (having the same referent), related in some manner across the dimension of time, much as the earlier relation schemas related schemas across spatial dimensions. Thus APPEAR, OCCUR, and GROWING-BRIGHTER are all relations of the same status as the spatial ones above.

This equivalence can be viewed from a somewhat different standpoint: A primitive relation is the output of some perceptual mechanism whose sole ability is to extract that particular type of relation from the external world. And since it is reasonable that one has perceptual analyzers which do this for spatial dimensions, its certainly logical for analogous analyzers to capture such relations across time, a dimension in which people are immersed as much as spatial ones.

In the notation below, the entities related by a relation schema have been expressed as "arguments" of the relation. By this I don't mean to imply that these entities are operated upon or processed by the relation, only that they must be ordered with respect to their connection to the relation. (In the representation examples later I will label each argument instead for reasons of clarity; these two alternatives are equivalent.)

Doesn't this essentially revert to the "slot" notation that I have been fighting against? Yes, but with an important distinction. There are relations whose arguments do not need to be ordered, e.g. NEAR. However, one cannot represent asymmetrical relations using only symmetrical ones as primitives. The proof is analogous to the proof that 'and' and 'or' are insufficient primitives for first-order calculus. Thus, order or labeling information is necessary. The key difference from schema-slot notation is that the labeling can never be known outside of the schema itself.

Virtually all of the relation schemas used in the models to follow are primitive. An non-primitive relation schema would represent the simultaneous occurrence of a group of relations, not necessarily between the same group of objects. Their possible existence, while necessary by the process assumptions below, is not needed for the purposes of the model here.



$$R_i \dashrightarrow R_i[\ R_{j_1}, R_{j_2}, \ldots, R_{j_i}\ ]$$

$$R_i \dashrightarrow R_i\$(\ a_1, a_2, \ldots, a_{k_i}\ )$$

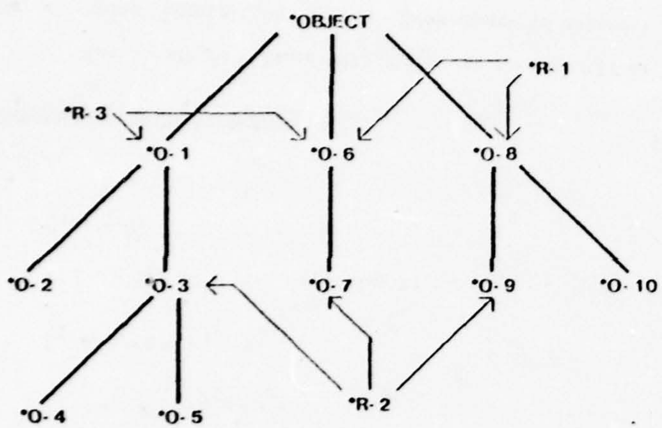
$$a_i \dashrightarrow O_i$$

$$a_i \dashrightarrow R_k$$

Elements from these object and relation sets combine to form the structures that are the complete internal representations of objects. The concept of an object would consist of an object schema, which would represent the simultaneous occurrence of its parts, which would be other object schemas. And these parts would represent the simultaneous occurrence of other object schemas, and so on, down to the level of primitive object schemas. Thus the basic structure of an object is represented by a "tree" of object schemas.

Relation schemas will connect various object schemas within the tree, usually schemas at the same level. (However, this is not a necessary constraint of the notation.) A sample structure representing an object is given in Figure 12. For examples of the structures of particular objects, see the schemas OBJECT-101 and TYPEWRITER in Chapter VII.

Class schemas. In this KRS these schemas represent what are often called in other systems "superset" concepts. With the process assump-




A sample structure of schemas representing an object

Figure 12



tions given below, and a couple additional premises, class schemas can also be used to represent all cases of what are usually called "generic" concepts.

The motivation for this second possibility is that one of the most difficult issues to deal with in a learning system is that of "type" or generic schemas, and what distinguishes them from token schemas, and how they can both be learned within the same system. I believe that, psychologically, the distinction between the two is actually quite vague, and that there are no real "type" schemas.

However, modeling an isolated domain with such a system is far too inefficient, for the characteristics of a generic concept would be spread throughout all the instances of the concept that the system had ever seen. Although it can be proven that such a system is possible, I'll retreat back to using distinct type concepts. In this simplified system, "class" schemas represent just concepts which are sets of other schemas:

$$C_i \dashrightarrow \{ x_{i_1}, x_{i_2}, \ldots, x_{i_{C_i}} \}$$

$$x_i \dashrightarrow O_i$$

$$x_i \dashrightarrow R_i$$

$$x_i \dashrightarrow P_i$$

$$s_i \dashrightarrow C_i$$

The class schemas in this example almost exclusively consist of objects.

Procedure schemas are like relation schemas in that they describe an interaction between two or more entities. However, while in

relation schemas, this interaction is across either space or time, procedures always include time and can include both. In addition, primitive procedure schemas share with primitive relation schemas, the need for ordered or labeled arguments.

There are at least two general types of non-primitive procedure schemas: those in which the subprocedures occur in a linear sequence (LS), with each beginning after the previous one ends, and those in which the subprocedures are simply a collection (PC). In this type, the executor of the procedure can execute the subprocedures in any order and with replacement. An instance of this second type is EDIT, modeled in the previous example.

$P_i$ ---> LS < $P_{j_1}$, $P_{j_2}$, ... , $P_{j_n}$ >

$P_i$ ---> PC < $P_{j_1}$, $P_{j_2}$, ... , $P_{j_n}$ >

$P_i$ ---> $P_i$$ ($a_i$, ... , $a_n$, time=t) ==> ($a_p$, ... , $a_z$, time=t+$\Delta$)

$a_i$ --> $O_i$

## Assumptions about Processes

The knowledge schemas within any one person will be constructed by a schema acquisition system in response to external and internal input, and the structure of the schema system developed will reflect the character of the processes of that acquisition system. These processes should be the same ones that will later operate on these schemas during complex learning.



The processes assumed for this schema acquisition system are of two classes:

Primitive processes:

(1) perception by senses

(2) encoding of time

(3) schema creation

Processes associated with each schema:

(1) automatic assimilation

(2) instantiation of other schemas

(3) modification of the contents of other schemas

These processes are defined in the assumptions within the following paragraphs.

(1) The acquisition system interfaces to a set of perceptual mechanisms, which gives to the system information in an already partially parsed form: Objects have been separated from background, but usually not identified. Most features and some subparts of the objects have been identified. Objects right at the primitive level, whose internal structures are not being considered, (such as the HOUSINGS in the example below) are always identified if they are known. Information such as motor feedback from a procedure is assumed to be encoded in a form which is manipulable like an object.

(2) The representation and encoding of the passage of time is crucial to the operation of the system. It is assumed that there is a primitive mechanism which processes objective time into a subjective form. This subjective time is divided up into an ordered series of

intervals, each of which themselves are timeless. Although the length will vary with what occurs during each interval, their mean time is assumed to be about 1/2 second. Some information, not necessarily accurate, is kept about the elapsed time between intervals. These time percepts will play a key role in the structure and modification of procedures, as well as the activation of causal mechanisms. In addition, it is assumed that the time interval in which any schema created or activated occurs, is added to that schema. The exact form is arbitrary for our purposes.

(3) In general, the simultaneous activation (or "occurrence") of two or more schemas within the same subjective time interval will be noted by the creation of a new schema which represents that simultaneous occurrence, and which will have as subparts the simultaneous schemas. This is the crucial construction mechanism within this acquisition theory, and accounts for the majority of the new schemas. In of itself, however, it can only be a very simple learning mechanism -- in some sense it simply "mirrors " the structure of the external world as encoded by its input. The complex learning to be modeled later occurs as a result of the combination of several mechanisms.

It should be noted that much of the input from the perceptual system described earlier is the result of this same mechanism: An object which is input is a schema which is simply the simultaneous occurrence of subpart and feature schemas which make up the object. If these subparts in turn have internal structures, those schemas represent the simultaneous activation of the parts of that structure, and so forth. It is assumed here that the same basic mechanisms operate throughout the



cognitive system, but that they blend at the perceptual end into more primitive perceptual mechanisms (e.g., line and angle analyzers and Gestalt processes).

(4) The basic process of and between schemas is a form of <u>activation</u>. Any schema can be activated directly or through part of itself. The activation of a schema actually consists of the <u>instantiation</u> or copying of a complete replica, or <u>instance</u> of itself. It is the instance which is active and which carries out the processes below. The schema and its copy are distinguished by the attached time marker of their initial activation (see (2) above). All activations and most resultant processes are assumed to occur in parallel.

Upon activation, a schema will begin a process of <u>automatic assimilation</u>. This process is defined recursively: If the schema is non-atomic, i.e. consists of a group of other schemas, then automatic assimilation is the activation of these subschemas. If a schema is atomic, then it must <u>find a match</u> for itself, from among the schemas within the current context. The current context is defined much as it was in Chapter V, and is simply the collection of all active schemas (hence it will include representations of the latest information input to the system).

The "finding of a match" to an atomic schema is assumed to be a primitive process: An atomic schema can only match another atomic schema representing the same concept. The success of this process is encoded as a primitive linking between the two instances. Failures in this process are noted and dealt with by "meta-schemas," to be described in (7).

(6) If an activated schema successfully completes automatic assimilation, then it activates its "parents," or those schemas of which it is a part or in which it plays a role. (This information is passed to the replica in the process of instantiation.)

(7) Finally, there are a class of schemas, called <u>meta-schemas</u>, whose structure has not been defined within the four sets of schema structures above. The reason for this is that I am not sure at this point what their best formal structure is. However, their associated process is defined here. Meta-schemas can be viewed as carrying out a process of assimilation very similar to other schemas, except that the process which initiates them is not input from the external world, but <u>information input from the failures of the automatic assimilation processes</u> defined above. Just as normal schemas are collections of other schemas representing patterns in the external world, meta-schemas, informally, are collections representing particular patterns of automatic assimilation failures. In addition, however, meta-schemas can direct further process among the normal schemas in order to attempt a recovery from the failures.

As a group these *additional* processes define the process of <u>complex assimilation</u>. They include the following: Activation of the higher-order (usually relational) schemas associated with a concept, even if assimilations among some of the parts of the concept have failed. The results from these activations can be used to activate new schemas, and "ignore" or deactivate currently active ones. The effect of this last process is to remove one or more schemas from a newly



instantiated schema structure, and is very often the crucial step in the learning of new structures. (An example of the operation of this process is given in the next chapter.) Here, I believe, is the actual source of what is learned in complex learning.

The formal *knowledge* representation described above still does not deal with several issues, and is vague on others, especially the structure of meta-schemas. However, I believe that it is a basis for a system which satisfies the criteria discussed earlier, and that the details can be worked out successfully. In addition, with the development so far, the character of the learning process at the level of this formal system can be described. The next chapter is an example of this application.

## VII: Typing on a Terminal: Modeling Complex Assimilation

The model below is an attempt, using the formal knowledge representation system described in Chapter VI and the processes which operate on it, to describe what appears to be a relatively minor learning inference made by the student: the learning of how to type on a terminal. I would like to accomplish three goals with this model:

(1) To demonstrate at a this *formal* representation level the character of the automatic and complex assimilation processes.

(2) To describe the form of higher-order knowledge, and show its role in complex learning.

(3) To push the model of the learning process to a *level of detail* that a computer implementation of the process is within sight.

The "terminal" is a machine whose purpose and operation are unknown to many of the students before the beginning of the experiment. Typical of the prior knowledge of terminals among these subjects is the example below:

MAN: When you are asked to do something, you should do it on the terminal to your left.

E: Why don't you define the word terminal as you know it. (E covers the screen)

S: A terminal to me means something like a bus station, but also something...can terminate.

E: When they say terminal to your left, what do you think they mean?





S: I have no idea.

E: OK, it means this (points to the terminal).

S: An interesting looking terminal (laughs).

Yet, given their (presumed) knowledge about typing on typewriters, and of the parts of televisions, they are able to infer that they can type on the machine, and that what they type is likely to appear on the screen.

I believe that this seemingly straightforward inference has quite a complicated process underlying it. The resemblance of this "terminal" object to either typewriters or televisions actually does not help in the possible identification of the object: Both of these known objects are clearly different from the unknown, and neither contains any information to explain or overcome the differences.

The new object also does not fit into the students prior knowledge of "typing." However, the "higher-order" structure of this procedure provides some information to overcome the differences: At this higher, "summary" level, the important events within typing are that the pressing of a particular key causes a particular character to appear. It is possible to carry over this same higher-order structure to a procedure for "typing" on this new object, and the learning has been effected. Note that the source information which allows the learning is within the higher-order structure of the procedure "type."

Is this actually how students carry out the inference? Strong evidence that at least some students are following approximately this reasoning is given by the protocol *segment below* (occurring before

beginning the computer editing instruction manual):

E: Do you know what this is? (points to the terminal)

S19: Probably a computer.

E: OK. Ah ... can you explain how this machine operates?

S19: What I think?

E: Yeah.

S19: I don't know how to operate this. But, um ... probably if it would write out something here, it would ask (unintelligible), I would ask it a question by typing it out here.

E: OK.

S19: How do you mean "operate this"?

E: OK. If you were to type something on this ...

S19: Keys?

E: Keys ... what would happen?

S19: Probably it would come out here (indicates the screen). But I don't know what would happen inside.

E: Fine.

S19: Somehow it would get transferred to a memory bank.

E: Do you have any idea why it would come out there?

S19: Because there's a screen. I just thought that ... by looking at the machine, since I saw the keys and then I saw the screen, I figured the screen it was for used ... like when you use a typewriter ... when you type on paper, usually it comes out so that you can see what you've typed... I figured that was what that was for.

E: I see. OK.

Note especially that her explanation for how she makes the inference centers around the procedure "type" and refers to other objects only in reference to this.



Finally, it is quite likely that many students do not have exactly the prior knowledge assumed here. (For example, a student may have seen a terminal in a movie or magazine, but not seen it in operation.) However, the purpose of the model below is to study the <u>character</u> of the assimilation process, and the role of higher-order knowledge in learning. The process which would occur in these cases would be slightly less complicated, but the character would remain the same.

<u>Modeling the Process of Automatic and Complex Assimilation</u>

An overview will be given first of the interpretive process that eventually leads to the learning of the knowledge of how to "type" on a "terminal." This will be followed by a description of the schemas involved in this process, in the notation of the knowledge representation system developed in the last chapter. Finally, a graphic description of the same interpretive process, but in terms of these schemas, will be presented.

In terms of the theory, the basic process which is assumed to occur is described in the following paragraphs:

(1) The subject receives from primitive perceptual mechanisms a visual parse (in terms of a group of schemas) of an object. The parts of this object, i.e. "screen," "keyboard," etc., have been identified by prior knowledge. However, the object as a whole has not.

(2) These parts activate other prior knowledge object schemas, that attempt to assimilate automatically the other parts of the object

-- that is, to interpret the object as an instance of themselves. The major contenders here are "typewriter" (activated by the "keyboard") and "television" (activated though the "screen").

(3) The automatic assimilations of these two contenders both fail, each because it has parts which are missing from the unidentified object, and because there are parts within the object which can't be assimilated.

(4) A procedure schema, "type," is also activated (because "keyboard" plays a role in part of the procedure). However, it's automatic assimilation also fails, since, among other reasons, there is no paper on which what is typed can appear.

(5) At this point, a process of complex assimilation begins, to try to recover from these failures. The meta-schemas directing this process will suggest several approaches. (For example, one is to compare the spatial relation schemas associating parts of the typewriter with the spatial relation schemas of the object.) However, the process that ultimately succeeds utilizes the higher-order structure of the TYPE procedure to extract the "essence" of the process of typing. This is used to force an assimilation of the object by the TYPE procedure, by ignoring parts of the procedure which aren't part of this essence. The parts of the procedure which remain will be the newly learned procedure for typing on a terminal.

The process in (5) can be described more specifically. As mentioned earlier, the essential events within the procedure TYPE are (a) the KEYPRESS of a key for a particular alphabetic character, and (b) the



APPEARance of a character. The essential relations between these two events are (a) that the first event CAUSEs the second, and (b) that the identity of the two characters in the events is the same. If these four pieces of higher-order information, and the parts of the TYPE procedure which are attached to them, can assimilate the new object, then the rest of the procedure will be ignored, and this interpretation of the object will be taken. This does happen, and the complex assimilation succeeds.

A Formal Representation of the Prior Knowledge Schemas Involved

The schemas below have been grouped according to the major object or procedure to which they belong, e.g. *OBJECT-101 or TYPE (proc). A brief description of some of the notational conventions is given first:

(1) One of the strings "r", "c", or "proc" in parentheses after a particular schema marks that schema as a relation, class, or procedure schema, respectively. All schemas not marked are object schemas. For example, *ENCLOSES(r)$-101 is a relation schema.

(2) An asterisk before a schema denotes that it is an instance. It will then also have a number after its name, preceded by a dash, in order to distinguish it from other instances of the same concept.

(3) If a schema is preceded by an exclamation point, it is a "partially instantiated schema." This is one without a direct referent within the external world, but with some slots partially or wholly constrained by its parent schema (that is, the schema in which it plays a role). In this case, the schema will have a number followed by a "p" after its name. (Partially instantiated schemas lie in the gray area inbetween types and tokens, and are one argument for considering the distinction as a continuum. Their use here, though, is not a theoretical stand, but a modeling convenience, since they have a relatively clear interpretation.)

(4) If a schema is preceded by no punctuation, it is a generic or type schema, and should be followed by a "-g".

(5) According to the representation system proposed in the last chapter, non-primitive object schemas simply represent the simultaneous occurrence of their subparts, that is, they do not have slot-names or even slots. The use of slots and the slot-names "part" and "feature" here is only for descriptive clarity, and plays no part in the model. They can be deleted without change in the description. For example, *OBJECT-101 would become *OBJECT-101 [ *RASTER-SCREEN-101, *KEYBOARD-101, *HOUSING-101, *ELECTRONIC-

MACHINERY-101 ].

(6) The slots labeled "parent-obj" and "roles" list the schemas in which a particular schema plays a part. These are also only for descriptive clarity, and can be simply deleted.

======================================================================

<u>Visual parse of the object</u>:
The schemas below are assumed to be generated by the perceptual processes which interface with this system. They have just "perceived" the object and recognized some of its subparts:

```
*OBJECT-101
        part: *RASTER-SCREEN-101
        part: *KEYBOARD-101
        part: *HOUSING-101
        part: *ELECTRONIC-MACHINERY-101

*RASTER-SCREEN-101
        parent-obj: *OBJECT-101
        role: *ABOVE(r)%-101
              *NEAR(r)%-101
              *ENCLOSES(r)%-101
        part: *DISPLAYED-CONTENTS(c)-101
                parent-obj: *RASTER-SCREEN-101
                element> *PICTURE-101
                           part: *SQUARE-101
                                   feature: *AREA-SIZE-101
                                                   element: *0.25sq.cm%-101

*KEYBOARD-101
        parent-obj: *OBJECT-101
        role: *ABOVE(r)%-101
              *NEAR(r)%-101
              *ENCLOSES(r)%-1011
        part: *KEYSET%-101
((KEYSET% has been set up as a primitive procedure which will return
a key in the 'key' slot below upon a request in the 'character'
slot below.  This may cause problems and be changed later.))
                character: !ALPHAMERIC-CHARACTER-10p
                key: !TYPE-KEY%-10p
                        feature: !NAME(c)-10p
                                     element: !ALPHAMERIC-CHARACTER-10p

*HOUSING%-101
        parent-obj: *OBJECT-101
        role: *ENCLOSES(r)%-101
              *ENCLOSES(r)%-1011
              *WITHIN(r)%-101
        feature: *COLOR(c)-101
```



```
                                 element: *BLUE%-101
*ELECTRONIC-MACHINERY%-101
        parent-obj: *OBJECT-101
        role: *WITHIN(r)%-101

*ABOVE(r)%-101
        upper part: *RASTER-SCREEN-101
        lower part: *KEYBOARD-101

*NEAR(r)%-101
        part1: *RASTER-SCREEN-101
        part2: *KEYBOARD-101

*ENCLOSES(r)%-101
        object: *RASTER-SCREEN-101
        framer: *HOUSING-101

*ENCLOSES(r)%-1011
        object: *KEYBOARD-101
        framer: *HOUSING-101

*WITHIN(r)%-101
        object: *ELECTRONIC-MACHINERY-101
        container: *HOUSING-101
```

======================================================================

<u>Television prior knowledge</u>:
Below are schemas associated with subject's prior knowledge of televisions:

```
TELEVISION-g
((g denotes a general schema))
        part: !RASTER-SCREEN-1p
((A number followed by a 'p' denotes a partially instantiated schema,
one without a direct referent within the external world, but with
some slots partially or wholly constrained by its "context", that is,
the parent schema in which it plays a role.))
        part: !HOUSING%-1p
        part: !ELECTRONIC-MACHINERY%-1p
        part: CHANNEL-SELECTOR-g
        part: !TUNING-KNOBS%-1p
        part: !ON-OFF-SWITCH%-1p

!RASTER-SCREEN-1p
        parent-obj: TELEVISION-g
        part: !DISPLAYED-CONTENTS-1p

!DISPLAYED-CONTENTS(c)-1p
        parent-obj: !RASTER-SCREEN-1p
        elements: { !PICTURE-1p, !MOVING-PICTURE-1p, !ALPHA-CHAR-GROUP-1p }
        feature: !LOCATION(c)-1p
```

```
elements> !WITHIN(r)%-1p
                container: !RASTER-SCREEN-1p
                obj: !DISPLAYED-CONTENTS(c)-1p

!HOUSING%-1p
        parent-obj: TELEVISION-g
            *HOUSING%-101  ((of *OBJECT-101))
```

((It is assumed here that all machine HOUSINGs are alike to a large extent, and therefore, within matching processes they will never be the cause of rejection. HOUSINGs will differ in size and shape, and this will be modeled by a primitive procedure which knows about HOUSINGs of particular machines, and which will return a degree of match based upon their degree of resemblance. This is a superficial approach, but it is necessary to avoid visual complexities, which aren't very relevant here.))

```
        [primitive process: Housings-PRIM]

!ELECTRONIC-MACHINERY%-1p
        parent-obj: TELEVISION-g
```

((As with HOUSINGs, all instances of ELECTRONIC-MACHINERY will match. However, here it is assumed that they do not even differ. I am modeling here an electronically very naive student.))

```
CHANNEL-SELECTOR-g
        parent-obj: TELEVISION-g
        part: CHANNEL-NUMBER(c)%
                elements: {2, 3, 4,...,80}
```

((A particular instance of a CHANNEL-SELECTOR is always positioned to a particular channel. This is modeled rather awkwardly as a particular "part".))

```
!TUNING-KNOBS%-1p
        parent-obj: TELEVISION-g
```

((This primitive concept is meant to represent some small number of tuning knobs, with the exact number unspecified. I haven't decided yet whether it is important to model unspecified amounts.))

```
!ON-OFF-SWITCH%-1p
        parent-obj: TELEVISION-g
```

((The role of this schema in turning a TELEVISION on and off would be expressed through a procedure concerning turning machines on and off, of which both this schema and TELEVISION-g would be arguments. This procedure in turn would be one of a collection of procedures involving watching TV (which would also include TV-DISPLAY below). However, the details of the WATCHING-TV schema are not central to the study here, and won't be modeled.

The purpose of this aside is to point out the nature of schemas such as TUNING-KNOBS and ON-OFF-SWITCH in the model here. Within a knowledge structure which is more complete within this area of the domain (e.g. includes WATCHING-TV), these schemas acquire some meaning through their roles within other schemas. However, one must cut a boundary somewhere, and in the process of severing, all the meaning that is left is that they are arbitrary parts of



TELEVISION, indistinguishable from a model in which they were called ROCK1 and ROCK2.
I believe that this loss or 'graying' of meaning is inherent to the process of cutting a boundary around what one wants to model. One needs to be careful to specify what meaning is left, but, with care, this graying need not spread inward from the boundaries.))

```
!NEAR(r)%-1p
        obj: !RASTER-SCREEN-1p
        obj: !TUNING-KNOBS%-1p

!ENCLOSES(r)-1p
                *ENCLOSES(r)%-1011
        object: !RASTER-SCREEN-1p
        framer: !HOUSING%-1p

!WITHIN(r)%-1p
        object: !ELECTRONIC-MACHINERY%-1p
        container: !HOUSING%-1p

!TV-DISPLAY(proc)%-1p
        source: TELEVISION-STATION%-g
        mediator: !ELECTRONIC-MACHINERY%-1p
        display-surface: !RASTER-SCREEN-1p
        output: !DISPLAYED-CONTENTS(c)-1p

!TV-DISPLAY(proc)%-2p
        source: TELEVISION-STATION%-g
        mediator: !ELECTRONIC-MACHINERY%-1p
        display-surface: !RASTER-SCREEN-1p
        output: !DISPLAYED-CONTENTS(c)-2p

!DISPLAYED-CONTENTS(c)-2p
        parent-obj: !RASTER-SCREEN-1p
        elements: { !PICTURE-2p, !MOVING-PICTURE-2p, !ALPHA-CHAR-GROUP-2p }
        feature: !LOCATION(c)-1p
                        elements> !WITHIN(r)%-1p
                                        container: !RASTER-SCREEN-1p
                                        obj: !DISPLAYED-CONTENTS(c)-2p

!APPEAR(s)%-1p
        obj: !DISPLAYED-CONTENTS-1p
        time: *NOW%-101
```

((NOW is actually a primitive procedure which computes a present time.))

```
!APPEAR(s)%-2p
        obj: !DISPLAYED-CONTENTS-2p
        time: *FUTURE%-101
```

((FUTURE is a primitive procedure which accepts any point in time further than the present.))

```
!FCIT(r)%-2p
```

```
(("FCIT" stands for "follows closely in time".  For a further
explanation of the nature of this schema, see the description
preceding another of its occurrences, in the higher-order structure
of the procedure TYPE.))
        first-event: !APPEAR(r)$-1p
        second-event: !APPEAR(r)$-2p

!DIFFERENT(r)$-1p
        obj1: !DISPLAYED-CONTENTS(c)-1p
        obj2: !DISPLAYED-CONTENTS(c)-1p
((DIFFERENT is a primitive relation which represents the noting
of an (unspecified) difference between two objects or events.))
```

=====================================================================

<u>Typewriter prior knowledge</u>:
Below are the prior schemas involved in the assimilation attempt of the 'terminal' object by the concept typewriter.

```
TYPEWRITER-g
        part: !KEYBOARD-1p
        part: !HOUSING$-2p
        part: !PLATEN$-1p
        part: !PRINT-MECHANISM-1p
        part: !INTERNAL-MECHANISM-1p
        feature: !MACHINE-TYPE(c)-1p
                elements: {MANUAL$-g, ELECTRIC$-g}
        feature: !BRAND(c)-1p
                elements> {IBM$-g, XEROX$-g, OLIVETTI$-g, etc.}

!KEYBOARD-1p
        parent-obj: TYPEWRITER-g
        roles:  the object of !ENCLOSES(r)$-2p
                the input2 of !KEYPRESS$(proc)-1p  ((see the procedure TYPE-g))
        part: !KEYSET(proc)$-1p

!KEYSET(proc)$-1p
((KEYSET is a primitive procedure which will return a key in the
((Note that although KEYSET is a procedure, its external appearance
is as a group of objects, the keys of the keyboard.
It would also seem that KEYSET would more easily be represented as
a class schema, but it is definitely not of that type: An
uninstantiated class schema represents a collection of instances.
An instance of such a schema contains just one element of this
collection, in contrast to KEYSET, where the instantiation will
contain a complete group.))
        parent-obj: !KEYBOARD-1p
        character: !ALPHAMERIC-CHARACTER-2p
                        elements: {!CHAR-A$-2p, !CHAR-B$-2p, ...}
                        feature: !CHAR-MODE(c)-2p
                                elements> !NONE$-1p
                        feature: !CHAR-CASE(c)-2p
                                elements: {LOWER-CASE$-g,UPPER-CASE$-g}
```



```
                        feature: !LOCATION(c)-2p
                                elements> !WITHIN(r)$-2p
                                        obj: !ALPHAMERIC-CHAR-2p
                                        cont: !MEMORY$-1p
        key: !TYPE-KEY$-1p
                feature: !NAME(c)-1p
                        element> !ALPHAMERIC-CHARACTER-2p

!HOUSING$-2p
        parent-obj: TYPEWRITER-g
        roles:  the framer of !ENCLOSES(r)$-2p
                the container of !WITHIN(r)$-3p
                the container of !WITHIN(r)$-4p
                the container of !WITHIN(r)$-5p

!PLATEN$-1p
        parent-obj: TYPEWRITER-g
        roles: the object of !WITHIN(r)$-3p

!PRINT-MECHANISM-1p
        parent-obj: TYPEWRITER-g
        roles: the object of !WITHIN(r)$-4p
                the mediator of !TYPEWRITER-PRINT(proc)$-1p  ((see TYPE-g))
        part: PRINTING-DIE-SET$-g
        part: TYPEWRITER-RIBBON$-g
                feature: !COLOR(c)-2p
                        elements> {BLACK$, RED$, BLUE$, GREEN$}

!INTERNAL-MECHANISM$-1p
        parent-obj: TYPEWRITER-g
        roles:  the object of !WITHIN(r)$-5p
                the mediator of !TYPEWRITER-GENERATE(proc)$-1p  ((see TYPE-g))

!ENCLOSES(r)$-2p
        object: !KEYBOARD-1p
        framer: !HOUSING$-2p

!WITHIN(r)$-3p
        object: !PLATEN$-1p
        container: !HOUSING$-2p

!WITHIN(r)$-4p
        object: !PRINT-MECHANISM-1p
        container: !HOUSING$-2p

!WITHIN(r)$-5p
        object: !INTERNAL-MECHANISM$-1p
        container: !HOUSING$-2p
```

=====================================================================

<u>TYPE prior knowledge</u>:
These schemas represent the student's prior knowledge involved with the

procedure TYPE. The higher-order structures of these schemas will eventually provide the key knowledge in assimilating the "terminal" object and inferring how to type on it:

((Note that the definitions of TYPEWRITER and TYPE(proc) as set up below are somewhat dependent upon each other.))

```
TYPE(proc)-g
linear-sequence%:
        !KEYPRESS(proc)%-1p
        !TYPEWRITER-GENERATE(proc)%-1p
        !TYPEWRITER-PRINT(proc)%-1p
```

---

```
!KEYPRESS(proc)%-1p
        roles: TYPE(proc)-g
        start-sig: !START-SIGNAL%-1p
        input1: !ALPHAMERIC-CHARACTER(c)-1p
        input2: !KEYBOARD-1p  ((see TYPEWRITER-g))
        mediator: !FINGERS%-1p  ((see !LEARNER-1p below))
        output1: !MOVEMENT(c)-1p
        output2: !MOTOR-FEEDBACK(c)%-1p

!START-SIGNAL%-1p
        roles:  the output from !START-A-PROCEDURE(proc)%-1p  ((see below))
                the start-sig from !KEYPRESS(proc)%-1p
                the object from !WITHIN(r)%-6p
        feature: !LOCATION(c)%-3p
                    elements> !WITHIN(r)%-6p
                                object: !START-SIGNAL%-1p
                                container: !MEMORY%-1p  ((see below))

!ALPHAMERIC-CHARACTER(c)-1p
        roles:  the input1 from !KEYPRESS(proc)%-1p
                the object from !WITHIN(r)%-7p
        elements:  ((all))
        feature: !CHAR-MODE(c)-1p
                    elements> ENCODED%-g
        feature: !CHAR-CASE(c)-1p
                    elements> {LOWER-CASE%-g, UPPER-CASE%-g}
        location: !LOCATION(c)-4p
                    elements> !WITHIN(r)%-7p
                                object: !ALPHAMERIC-CHARACTER-1p
                                container: !MEMORY%-1p

!MOVEMENT(c)-1p
        roles: !KEYPRESS(proc)%-1p
        elements> !PUSHED-DOWN(r)%-1p
```

((The relation is between a before-state of the TYPE-KEY below and an after-location. However, for clarity, just the object and the location changes will be specified.))

```
                    obj: !TYPE-KEY%-1p  ((see TYPEWRITER-g))
                    loc-change: -1cm%-g
                    time: !AFTER(r)%-1p

                    form> MECHANICAL-g
```



---

```
!TYPEWRITER-GENERATE(proc)-1p
        roles: TYPE(proc)-g
        input: !MOVEMENT(c)-1p   ((see above))
        mediator: !INTERNAL-MECHANISM%-1p  ((see TYPEWRITER-g))
        output: !SIGNAL%-1p

!SIGNAL%-1p
        roles:  the output from !TYPEWRITER-GENERATE(proc)%-1p
                the input1 from !TYPEWRITER-PRINT(proc)%-1p
        feature: !FORM(c)%-2p
                    elements> the elements from !FORM(c)%-1p ((TYPEWRITER-g))
```

((This is a slightly odd construction.))

---

```
!TYPEWRITER-PRINT(proc)%-1p
        roles: TYPE(proc)-g
        input1: !SIGNAL%-1p  ((see above))
        input2: !PAPER-SHEET%
        input3: !LOCATION(c)-1p
        mediator: !PRINT-MECHANISM-1p  ((see TYPEWRITER-g))
        output1: !ALPHAMERIC-CHARACTER(c)-2p
        output2: !MOVEMENT(c)-2p

!PAPER-SHEET%-1p
        roles: the input2 of !TYPEWRITER-PRINT(proc)%-1p
        feature: *FLAT-SIZE(c)-101
                    element: *(8 1/2 x 11 in.)%
```

((This is actually prototypical information and must be changed if a mechanism for prototypes is developed.))

```
!LOCATION(c)-3p
        roles: the input3 of !TYPEWRITER-PRINT(proc)%-1p
        elements> !TWO-DIMEN-COORD(c)-1p
                    elements: ((all, primitive))
                    feature: !LOCATION-UNIVERSE(c)-1p
                                element> !PAPER-SHEET%-1p

!ALPHAMERIC-CHARACTER(c)-2p
        roles:  the output1 of !TYPEWRITER-PRINT(proc)%-1p
        elements> the elements of !ALPHAMERIC-CHARACTER(c)-1p
        feature: !CHAR-MODE(c)-2p
                    elements> !PRINTED%-1p
        feature: !CHAR-CASE(c)-2p
                    elements> the elements of !CHAR-CASE(c)-1p
```

```
feature: !LOCATION(c)-4p
                elements> the elements of !LOCATION(c)-3p

!MOVEMENT(c)-2p
        roles: the output2 of !TYPEWRITER-PRINT(proc)%-1p
        elements> !TYPEWRITER-INCREMENT(r)%-1p
((This relation is to represent the change in the printing location
generated by typing a character.  (As in the case of PUSHED-DOWN
it is technically the relation between two states.)  The actual
change will vary somewhat with the character pressed, e.g.
Carriage-Return.  These complications will be handled at the
primitive level.))
                        obj: !LOCATION(c)-4p  ((above))
                        time: !AFTER(r)%
```

---

```
((Below is the higher-order structure of TYPE-g.  Chronologically,
the schemas below would have been created before the schemas
above, during the development of the learner's knowledge of
typing.  These higher-order schemas, along with the meta-schemas,
would then have constructed the linear sequence of procedures
above as an explanation.  However, this process is assumed
to have occurred before the initial state.))
!APPEAR(r)%-1p
        object: !ALPHAMERIC-CHARACTER(c)-2p  ((above))

((The schema below will be used to represent the time spacing
between all pairs of events which occur within about one
second of each other.  The theory proposes that humans assume
that events which follow each other within that time frame
are related and probably causally related unless given
information to the contrary.  The occurrence of such event
pairs should automatically initiate an explanation process,
which, in the case of TYPE, would eventually generate the
procedure sequence.
        In a more precise model, finer gradations of time would
be noted.  This could be modeled by making the schema FCIT
a class schema whose members are FCITs of different time
durations.  This is more precise than is needed here.))
!FCIT(r)%-1p    ((for Follows-Closely-In-Time))


((The schema below would be generated during the development of
TYPE as soon as no prior knowledge or information to the contrary
within the context could be found.))
((Note that the explanation process would just begin then, for
the reason for the causing would have to be found.))
!CAUSES(r)%-1p
        causer: !KEYPRESS(proc)%-1p
```



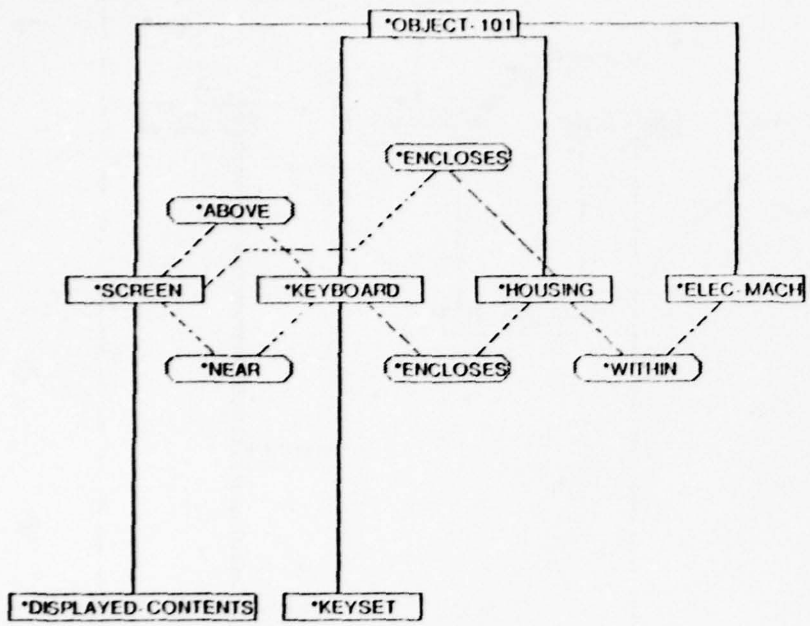
```
        result: !APPEAR(r)%-1p

((One of the first conclusions to be drawn from this explanation
process is the (salient) occurrence of instances of
ALPHAMERIC-CHARACTER as roles within each of the events.
This knowledge is encoded in the schema below.
The drawing of this conclusion and thus the creation of the
schema must occur through the operation of meta-schemas.  The four
schemas above, FCIT, KEYPRESS, APPEAR, and SAME-CONCEPT, together
approximate what might be expressed in English as:
"Pressing a key with a particular character on it caused
that same character to appear in printed form on the paper
held by the machine (typewriter)."))
((Note that during the initial development, all of these
schemas would be true instances (referring to the first typewriter
seen) and not just partially instantiated ones as used here.))
!SAME-CONCEPT(r)%-1p


((This correspondence of ALPHAMERIC-CHARCACTER instances would
then be used along with prior knowledge about printing to
continue the explanation.  In addition to the procedure sequence
described here, other "intermediate" structures like SAME-CONCEPT
would be created and are actually part of the final complete
structure of TYPE, but they aren't needed in what follows.))
```

---

## A Description of the Assimilation Using Activation Diagrams

It is possible to specify the activation and automatic assimilation processes of each of the above schemas as they participate in the overall assimilation of the object. However, this serves to muddle the important processes and points that I would like to get across. Therefore, I will describe the process at a slightly higher level, using the "activation diagrams" in the accompanying figures to sketch in the details.

Figure 13 is a sketch of the schema structure of the "terminal" object, *OBJECT-101. One automatic assimilation process is initiated by a subpart of this object, *RASTER-SCREEN-101 activating a new instance




Schema structure for *OBJECT-101

Figure 13

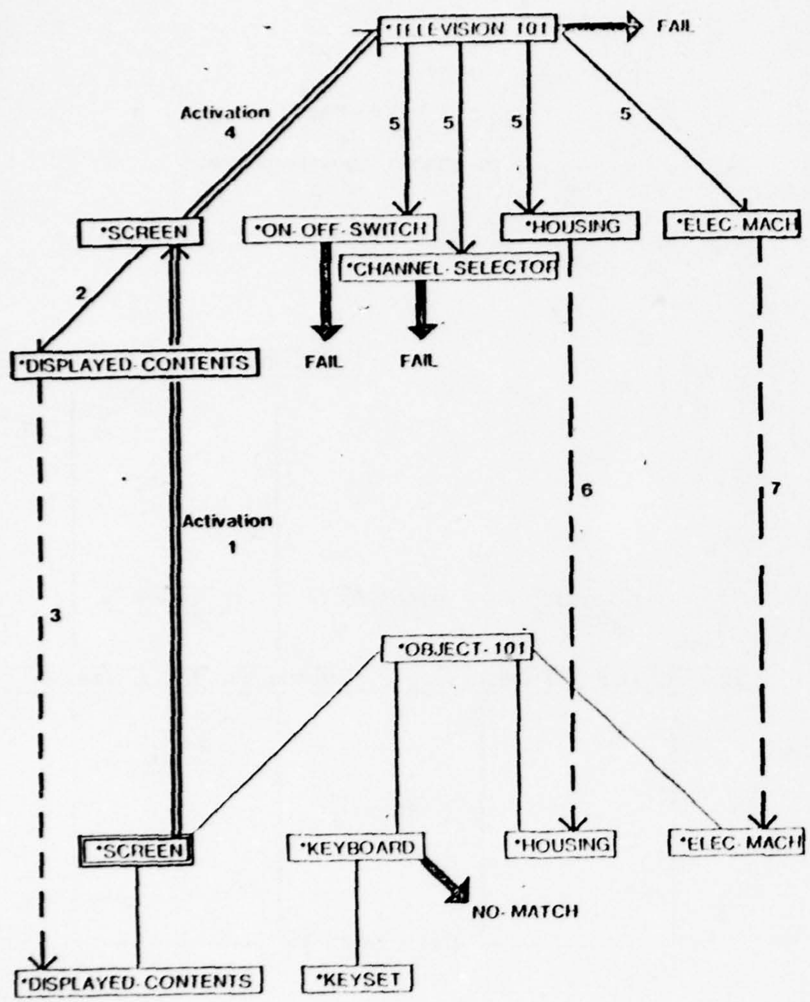


of television (to be called *TELEVISION-101).

This new schema then attempts to assimilate the other parts of *OBJECT-101. This assimilation process is modeled in Figure 14. The initial activation (marked '1') of *TELEVISION-101 occurs through one of its parts, *RASTER-SCREEN. Before activating its parent, this RASTER-SCREEN part must ascertain that its own subparts can assimilate parts of OBJECT-101. This succeeds, as noted by the path marked '2' and '3'. Activation can then proceed to *TELEVISION-101, and the same process of assimilation, but carried out at one higher level, occurs. (As described in Chapter VI, an assimilation process consists of activation of the subschemas of which a schema is constituted, unless the schema is primitive. In this case, it will attempt to find another instance of itself within the initiating schema, in this case *OBJECT-101.) In this diagram, successful assimilations are marked by dotted arrows linking the paired schemas, and unsuccessful ones marked as arrows pointing to a "Fail". For an automatic assimilation by a schema to succeed, the assimilations of all of its subschemas must succeed. In this case, the assimilation fails.

Figure 15 diagrams a similar attempt by a new instance of type-writer, *TYPEWRITER-101, to assimilate *OBJECT-101. This process was initiated though *KEYBOARD-101. This assimilation attempt also fails.

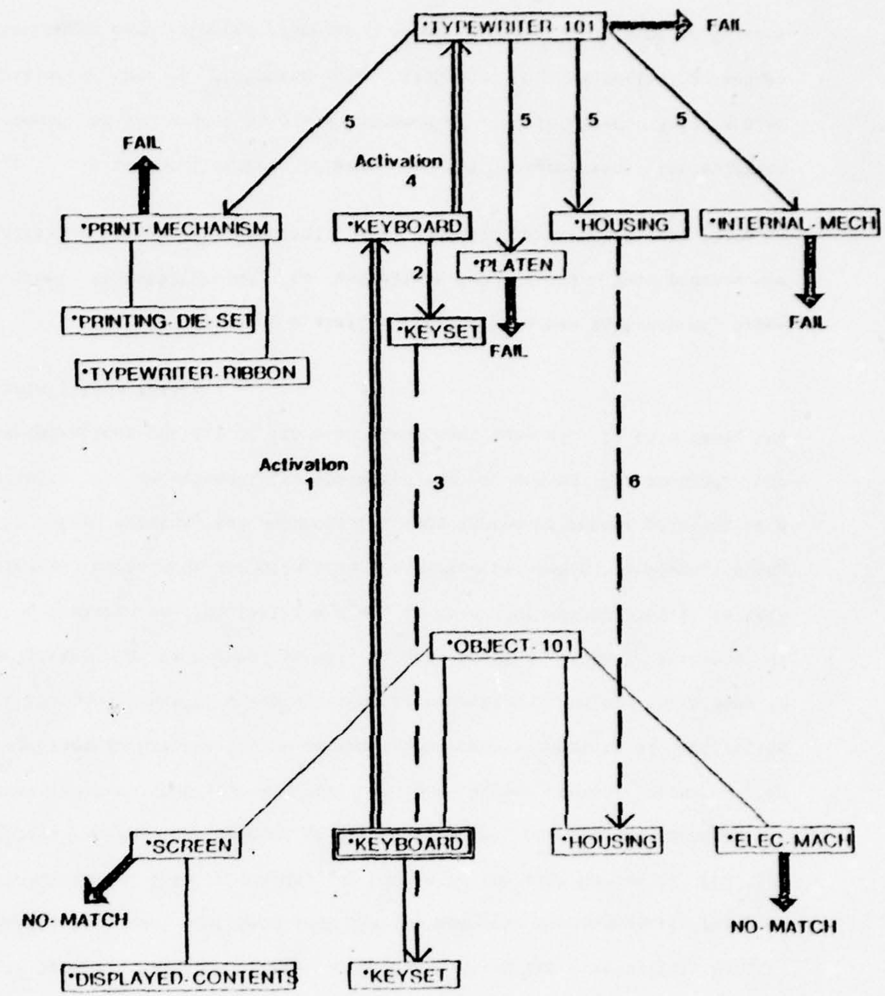
In the next diagram (Figure 16 the unsuccessful assimilation process of the object by a new instance of TYPE is given. This is also initiated through *KEYBOARD-101. Although the assimilating schema structure here is a procedure instead of an object, the process is very




Assimilation attempt of *OBJECT-101 by *TELEVISION-101

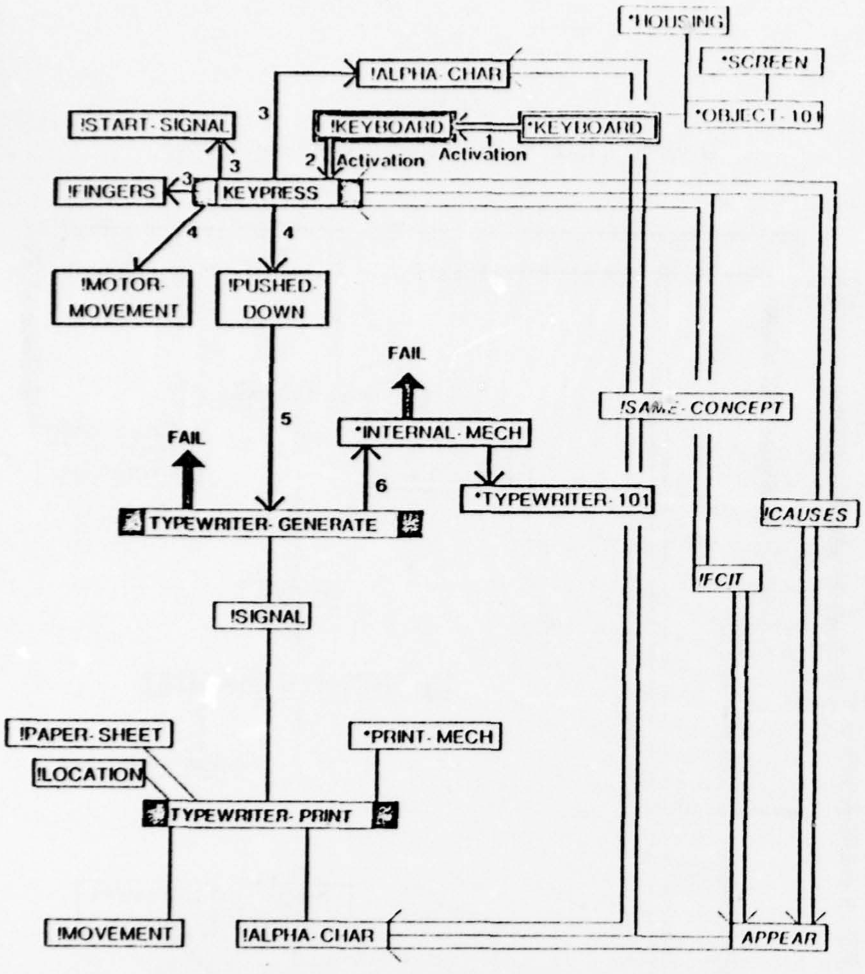
Figure 14






Assimilation attempt of OBJECT-101 by *TYPEWRITER-101

Figure 15




First assimilation attempt of OBJECT-101 by TYPE

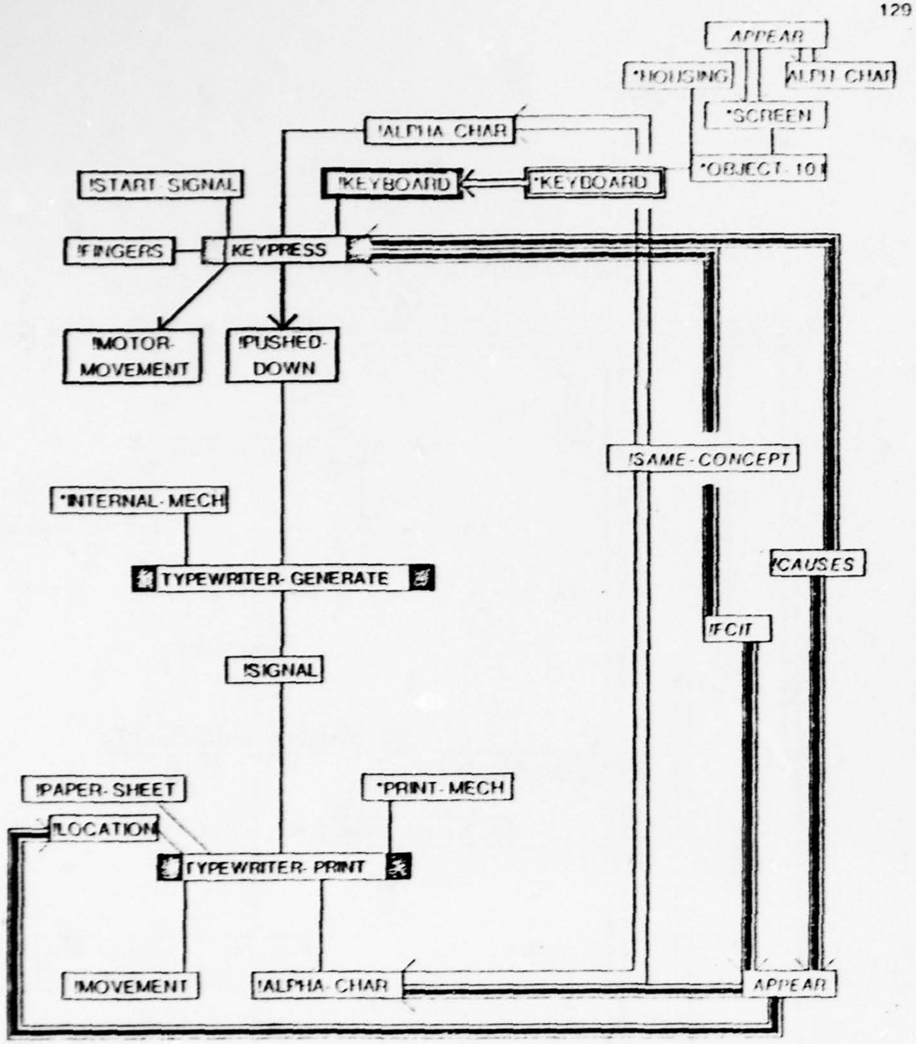
Figure 16



similar. Again the numbers along the paths indicate the approximate time course of the process. The first subprocedure within TYPE, KEYPRESS, is able to successfully complete assimilation. However, assimilation attempts fail when the process is carried to the next subprocedure, TYPEWRITER-GENERATE. Thus, the <u>automatic</u> assimilation by TYPE of *OBJECT-101 also fails.

Finally, the last three diagrams, Figures 17, 18, and 19, trace the process of complex assimilation by TYPE of *OBJECT-101. This process is initiated after the failure of the automatic assimilation, and is based upon the successful assimilation by the subprocedure KEYPRESS, and its salient role within the higher-order structure of TYPE. As a first step, just those parts of TYPE which play roles in this higher-order structure (see Figure 17) are utilized in an attempt to assimilate parts of *OBJECT-101. This includes the subprocedure KEYPRESS and the APPEARance of the ALPHA-CHAR within TYPEWRITER-PRINT.

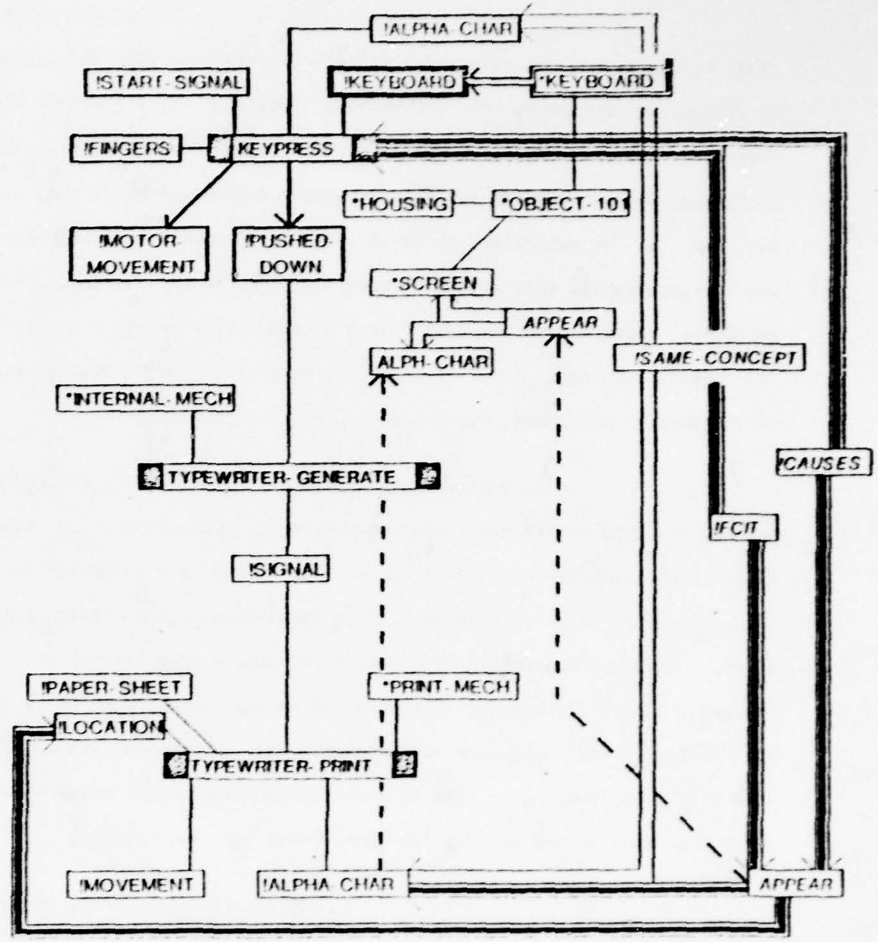
Since KEYPRESS as already successfully completed assimilation, only APPEAR must assimilate *OBJECT-101 parts. This will depend crucially on a piece of information gotten from the unsuccessful assimilation of *OBJECT-101 by *TELEVISION-101: *SCREEN from *TELEVISION-101 has the property that various forms of pictures can APPEAR on it, and one *form* of these pictures are ALPHA-CHARs (see the schemas for TELEVISION given in the previous section). As a result of this earlier assimilation attempt, all of this information is inherited by *SCREEN of *OBJECT-101, and the relevant part is shown attached to *SCREEN in Figures 17 and 18.




Complex assimilation of OBJECT-101 by TYPE (step 1)

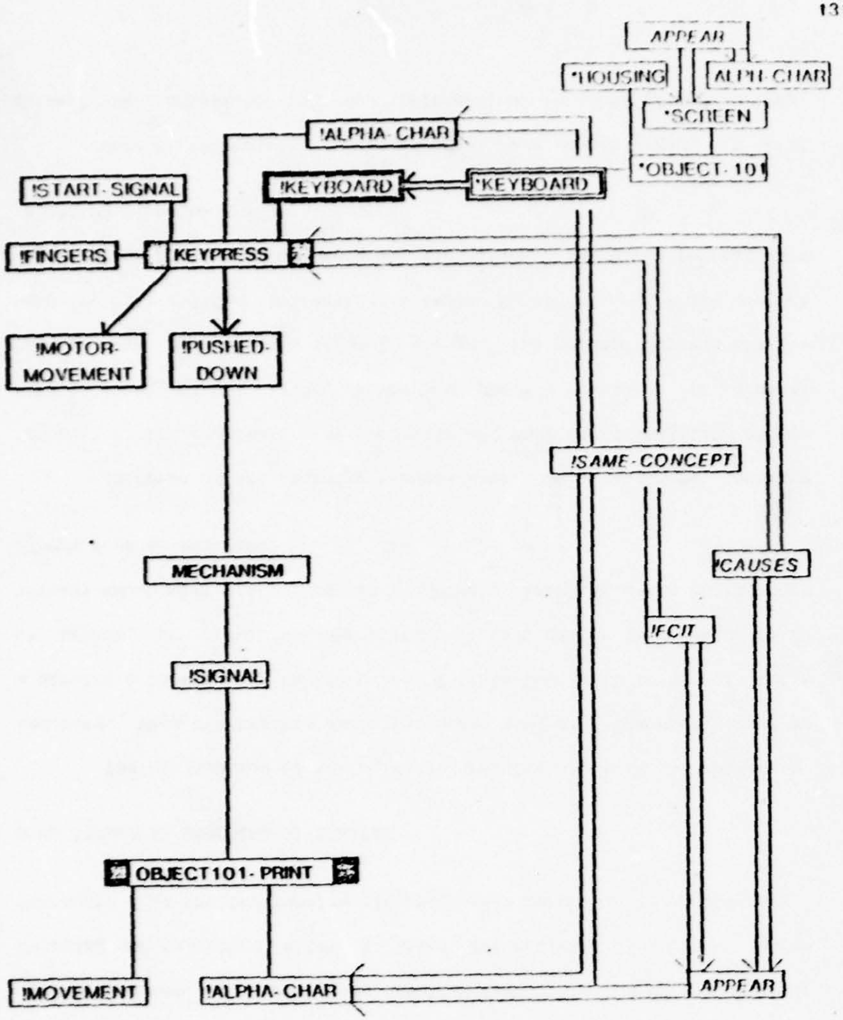
Figure 17






Complex assimilation of OBJECT-101 by TYPE (step 2)

Figure 18




Complex assimilation of OBJECT-101 by TYPE (Final step: new procedure)

Figure 19



This information will allow the APPEAR within TYPE to assimilate a part of *OBJECT-101, where the SCREEN within *OBJECT-101 now takes the role of the LOCATION on the PAPER-SHEET within TYPEWRITER-PRINT. Thus, using just schemas important within the higher-order structure of TYPE, a partial assimilation can be made of *OBJECT-101. The complex assimilation is completed by meta-schemas (not shown) which suppress or delete information not playing roles within the higher-order structure (such as TYPEWRITER-GENERATE). In addition, more general purpose procedural schemas are used to replace this deleted information and complete the proposed new procedure for how to "type" on *OBJECT-101. A sketch of this new procedure is shown in Figure 19.

## VIII: Directed Analogy Learning

This example considers an application of the theory to a learning phenomenon at a much less detailed level: the use of analogies in learning and teaching. After a brief theoretical discussion, three instances from the instruction situation used here will be considered.

### Some Issues in Learning by Analogy

The utilization of analogies is prevalent in both learning and teaching. When one teaches about the human mind as a computer, chess as a strategic battle, learns the rules of table tennis in terms of those in tennis, or picks up a new computer text editor in relation to an already known text editor, one is teaching or learning using an analogy of one form or another.

In terms of the learning process theory described here, analogy learning is the combination of automatic and complex assimilation occurring at a more global level, involving whole structures of schemas rather than single ones or small groups. The process depends heavily upon the higher-order similarities between the structures, which govern the analogy process at the more detailed level. There the process here is called directed analogy learning.

Once an analogical structure has been established, it will attempt to assimilate all new information in terms of the analogy.





Thus, this structure, and the interpretation of the instruction which results, once established is very difficult to unseat. Data which can't be assimilated, but which don't contradict the structure directly, are likely to be simply ignored. Even contradictory data sometimes need to be quite salient before a reorganization or rejection of the structure occurs. (Very similar events were observed in Chapter II, where hypotheses about "3p", once set up, assimilated all new information and were difficult to reject. The only difference was that the original structures were developed through possibility chains and not as an analogical structure.)

This phenomenon has implications for various levels of analogies which occur in both learning and teaching. If an analogical knowledge structure is very bad, that is, if there is very little match between it and the knowledge it is trying to assimilate, it will provide very little help for the learning process: It won't assimilate much new knowledge and contradictory information will tend to come along quickly. In addition, it is likely not to be activated in the first place.

As the analogical structure becomes more similar, it helps more: Some data will be assimilated and organized as they come in. Also, when the analogy fails and the reorganization-complex assimilation occurs, much of the structure may be recovered in the restructuring. If an analogical structure is identical, or nearly so in the sense that no significant structure is different, it will be of much help and the process will proceed smoothly (mostly through just automatic assimilation). In effect, very little needs to be learned, for it is all known.

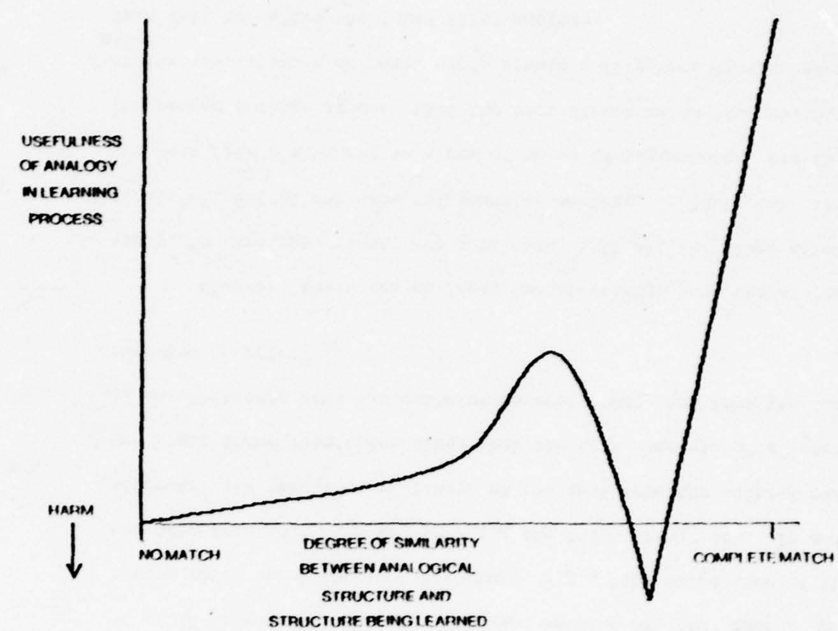
However, if the analogical structure is slightly less than perfect, with one or two significant differences, the analogy becomes insidious: it appears to the student to help learning but in fact can be quite harmful. (See Figure 20. This figure is to be taken just as a sketch of a trend). The structure assimilates most new knowledge correctly and becomes firmly established. When the significant different piece comes along, it is often rejected, even if it is contradictory or causes many problems.

Note that this phenomenon is particularly harmful when the analogy is being used only by the student, unknown to the teacher. If the teacher introduces the analogy, or is aware that the student is using it, he or she can point out the significant difference. The analogy can often be quite useful the, but <u>the difference must be very carefully pointed out</u> (see Figure 21).

This introduces an second phenomenon: Because of the automatic nature of the activation and assimilation processes, possible analogical structures can be activated and established by the student when the teacher had no intention of introducing them. To the extent that they are good analogies, this does not harm and is probably useful. However, if they are in the nearly perfect category, they can be quite harmful. And the teacher is often not aware of the cause -- the student's behavior may seem quite bizarre if not analyzed carefully.

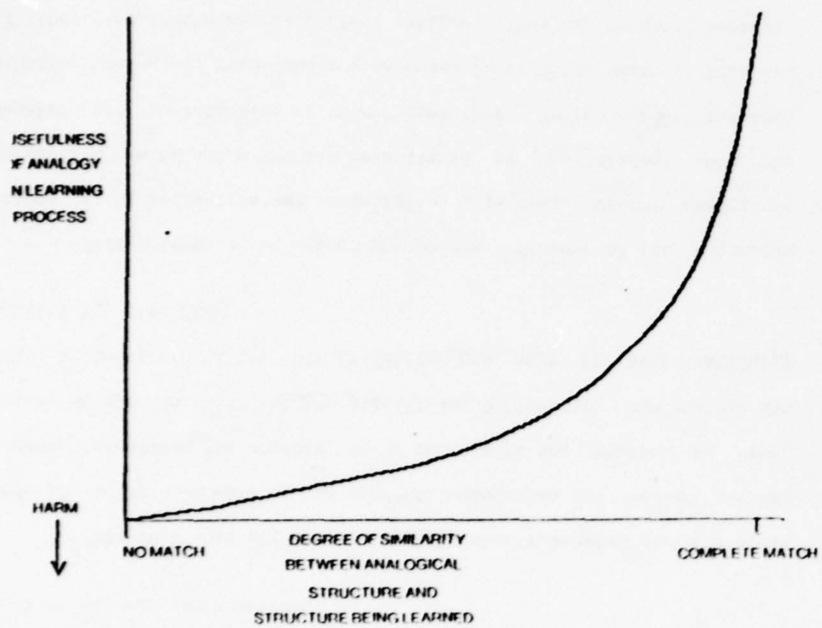
In the examples below from the instruction situation used here, we'll examine the effects of this analogy process. In all cases the analogies used were introduced by the student, and would have been unk-






Only the learner knows of the analogy

Figure 20




**Learner and teacher both know of the analogy**

**Figure 21**



nown to the teacher except through a protocol. In all cases, the effect of the analogy introduction is at least slightly harmful.

Case 1: The Structure of the Text File

The actual structure of the text file on which the students are editing is never explicitly described to them in the instruction manual. It would appear that the print command section of the manual provides strong clues about the text structure, e.g., "The third line of the text has been printed," "Now try printing the fifth line," etc. In addition, students see particular lines of the text from the editing examples. One might infer from these clues that the text consists of a series of lines, that each line has contains several words, and that the lines are numbered in order.

However, there are at least two structures that the student can apply in analogy, that are consistent with all the clues given. The first, which I'll call the "pigeonhole" analogy, is that the lines of the text file are stored in a row of boxes or pigeonholes, one to a box. The second analogy assumes that the text structure is like having lines of the text written on index cards within a file, one line to each card. This will be called the "card file" analogy.

In neither case does the student have to use the specific objects within these analogies. They must simply have the structure underlying the analogy as prior knowledge, and use this structure to interpret the text file.

Within the print command section, either of these could be used as a structure with which to interpret references to the "text." And, in fact, both of them are compatible with the p command: In the case of "pigeonholes," to print a particular line would mean to fetch the line from the appropriate box and display it. Similarly, to print could mean to pull out and display a particular index card.

However, the two analogies predict different results in the delete command: To delete a line from a pigeonhole leaves an empty box, and attempts to print that line later will produce a blank, or perhaps a question mark. In deleting a line from a card file, one removes the card, and an attempt to print that numbered line will produce the line after the line. (The second analogy is in fact correct.)

Thus, depending upon which analogical structure is used the student will either have few difficulties interpreting the effects of the delete command, or be unable to understand the results of future commands after a line has been deleted. Most subjects appear initially to choose the pigeonhole analogy, and this usually causes some problems. Many, however, are aware of the other possibility, and are able to recover eventually.

However, the pigeonhole hypothesis appears to be quite attractive, and is often not given up immediately even in the face of contradictory evidence. For example, a student (who had very little trouble with the p command) produced the following protocol segment. This occurred after the student had deleted the fourth line and printed out the entire text, showing the fourth line gone with no blank line left:



E: OK, let me ask you a question right now. What happens when you type 4d, I mean 4p?

S: 4p? Right now? It would give me a question mark.

E: OK. What would happen if you type 4d right now?

S: (long pause) It might give me a question mark again. I'm not sure, because it's already deleted. (S winces)

E: Why did you wince there?

S: Oh, no, no. No it wouldn't. OK, if I did 4d now, it would delete the fourth line of which it is now. Because I guess the lines all change, so it would delete "of knowledge ... become crucial" [the new fourth line].

E: Why did you decide that now?

S: Because all the lines changed. They moved up. They used to have a different number ... assigned to them now. *(pause)* Well no, maybe not. ... No, I don't say that. Each of the lines has still their same number. If I typed up the fourth line... no I don't know about the fourth line... If I typed up the fifth line ... it would give me this fourth line here. [of knowledge ..]

Even if the subject understands the effect of the delete command, the pigeonhole analogy can resurface again in the append command: *Some subject worry* that they won't be able to add more than one new line of text between two old ones because there won't be room.

The teacher could take advantage of the automatic use of analogies here by specifying the correct analogy to use, i.e. that the text is like a numbered card file with one line on each card. In *essence*, the student then automatically applies the correct structure in interpreting references to the text file. (The problem is often that the teacher does not know of the analogies being applied.)

Case 2: Prior Print Definitions as Analogies

The protocol segments in Chapter II suggested that some students interpret "Now print the fifth line of the text" as a request to make up a fifth line and type it in. This has been earlier interpreted as prior knowledge about print assimilating information from the instruction. Within the context of this chapter, it can also be viewed as the application of an analogy: the student's prior definition of print as write, to the request within the instruction sentence.

This analogy is interesting because it is an example of a "close" analogy (i.e., the definition is close enough to what is really meant to appear to be useful). Yet its use misleads the student's attempts at interpretation.

This suggests that one could perhaps produce fewer problems if one substituted even a nonsense word for "print," since then the damaging analogy could not be applied. This experiment was tried on several subjects: the word "gordol" was substituted for every use of print within the manual (the p command now became the g command for consistency).

The results were mixed, usually causing a greater confusion at the beginning of the "gordol" section of the manual, but sometimes leading to fewer misinterpretations later. This is not unexpected, for in taking away the analogical structure, one also sacrifices the use of the similar parts of the structure.



One student's summary of the g command upon completion of the gordol section is given below. The description appears to be very close to the structure of the print command. Note, however, that the word "print" does not occur as part of the explanation: The prior knowledge of this word does not fit with the structure of the command.

E: OK, um, before we go on, um... can you summarize the g command? ... the gordol command?

S: Uh, the g command seems to be a way of, ah, recalling... specifically line by line, of a um, of a line that's been fed into the computer, at one time ... ah ...

E: OK ...

S: It can be called individually, or a whole text.

E: Fine, ah ...

S: But only what it has been fed, I mean obviously because,... I think it was 4 and 5 are the same, you know ... (unintelligible) ... only as efficient as the operator who put it in.

E: OK, um, a couple of questions: When you say "recalling", what do you mean? ... In what sense is it being recalled? S: Well, it must have some type of a memory ... ah .. system, where obviously you or somebody had to put this into the machine at some other time, and then it just .... I don't know how it would be done ... I mean somewhere it would somewhere register when you put in, but then it brings it back by ... referring specifically to the g command you get this text.

E: OK, fine. If you wanted to ... ah, gordol the fifth through seventh lines, how would you do it?

S: Alright, should I do it, or ...

E: No, just tell me.

S: I'd push 5,7g RETURN.

Case 3: Computer Text Editor as Dictation Secretary

The perception of the computer text editor being learned in analogy to a human secretary taking dictation appears to be developed

spontaneously by most subjects (it is never mentioned in the manual or by experimenter). This phenomenon is probably quite reasonable: One common task of most people's "prototypical secretary" is probably editing letters or manuscripts. Also, in analogy to a dictation situation, the p command is like the dictation giver saying "Read back to me that last line again." Similarly, the d command is analogous to "Cross out the first line of that paragraph."

The analogy is probably very useful to the student in setting up, and, in fact, the text editor does serve a similar function. However, it breaks down in a subtle way in relation to the a (or append) command.

The basic structure of the append command is as follows: To append additional text lines after some line in the current text, you type that line number and "a", followed by RETURN. At this point, every line of text that you type will be added to the current text after that line. When you are finished adding, you type a line with only the word "end" and type RETURN. You can then type other commands to the editor. For example, to add

*It is not a simple activity.*

after line 7 of some text file, you would type

```
7a
It is not a simple activity.
end
1,$p
```

And the last command would print the entire text, including the line "It is not a simple activity." after line 7. However, what is often typed



by the students in the experiment for this exact task (see Appendix B) is:

```
7a
It is not a simple activity.
1,$p
```

The line containing only the the word "end" is *missing*.

This phenomenon has been named the "append mode trap": The result is that the command "1,$p" has been appended to the text file (the subject is "trapped" in the append "mode"). In addition, once it has occurred, subjects do not realize their error, and have an extremely difficult time determining the problem. (Many in fact never recover without hints.) This is despite being explicitly told in the manual that, to stop appending, one needs to type a line containing only the word "end." (The exact phrasing here is not important; variations of this wording have been tried with the same results.)

However, if one assumes that the student is assimilating all new information in terms of an analogical "secretary-taking-dictation" structure, the behavior becomes quite reasonable: The string "1,$p" is "obviously" a command; therefore the computer text editor (secretary) should take it as such and not continue appending to the text. In the analogous situation, if one were dictating to a secretary and then said, "Now read back what I've said so far", a secretary would not add this to the dictation.

Finally, this analogy in assimilating each instruction statement will reject reject statements which don't fit into the structure. Therefore, subjects in reading

The only way to stop appending is to type a line containing only the word "end"

essentially reject this statement since it doesn't fit into the analogy. In fact, some subjects after falling into the append trap were allowed to reread this section of the manual, and still did not notice this sentence.

What is the solution to getting subjects to avoid the append trap? It is possible to phrase the manual to get subjects to avoid this. However, several rephrasings that the teachers thought surely would solve the problem, did not. In addition, even if the problem is avoided, students still have difficulty in this section. A perhaps more satisfactory and *general solution*, *redesigning* the editor, is discussed in the next chapter.

## IX: Conclusions

This thesis has tried to construct a theory of the process underlying complex learning, and to apply this theory to modeling the learning process at various levels of detail. The major points of the thesis are:

(1) That the process of learning complex material, even in cases where it appears to go quite smoothly, is a much more complicated and meandering process at the micro-level.

(2) That it is still possible to model this *complicated process*, and that it reflects largely, and in often surprising ways the prior knowledge structures of the student interacting with the information implicit within the instruction.

(3) That the learning mechanisms involved are largely simple and automatic: the complexity of the learning process is a reflection of the prior knowledge structures (and the information being taught). The basic assumption of these learning mechanisms is that all instruction being provided already resides within prior knowledge.

(4) That the key information within the prior knowledge structures which allows complex learning to take place is the similarity between higher-order structures.

(5) Finally, in order for the learning model to have any significance, it is important that the prior knowledge structures built can be created in a reasonable (and psychological) fashion by these same learning mechanisms. Therefore, a careful analysis of the knowledge representation system to be used is necessary.

This learning theory is not a prescription for teaching. However, teaching and learning are integrally tied within this theory: As we have seen in the examples, the course of the learning process is highly dependent upon the choice of even exact words within the instruction. Therefore, to the extent that the theory can model this process, it should be able to tell the teacher exactly how to guide it. Guidance cannot yet be provided at this level; however, several somewhat more general conclusions are suggested.

First, because all instruction appears to be assimilated as though it had the structure of prior knowledge, teaching by good analogies known to the student can be highly efficient: It provides a structure to the student with which to easily interpret new information. While this is in some sense is well-known, the application of the method must be done much more carefully than is perhaps usual: The structure of the analogy must be analyzed in detail, and the presentation well planned out. Any discrepancies between the analogy structure and what is being taught must be carefully emphasized (Chapter 8 demonstrates the effects of not doing so).



However, given these constraints, the method has surprisingly wide applicability. For example, within the computer text editor here, the structure of the text file can be explained in analogy to a file of index cards, and each of the commands can be explained in terms of this analogy. The inherent structure of a card file circumvents, for example, the worry of several students that there is no "room" between lines 7 and 8 to append several lines. In conjunction with this, one can explicitly use the secretary analogy, but carefully pointing out the differences.

A second implication is an explanation of why there is a huge advantage in the use of <u>examples</u> in teaching. Many of the problems described in Chapter II earlier can be characterized by the teacher's *inability to foresee* the student's interpretation process of verbal descriptions within the instruction.: In essence, the teacher in describing the domain to be taught, has translated it into a form which no longer describes it as far as the student is concerned. The use of examples avoids this by skipping the translation process: The teacher is forced to teach the knowledge domain he or she intended to because the example (by definition) has the structure of the domain.

Third, it is clear that a thorough analysis of exactly what the structure is of the knowledge domain that you are teaching (as in Chapter II), often goes a long way towards guiding clear instruction. Just as important, it makes an analysis of the student's problems much easier. This underlies the elegance and promise of the approach taken in SOPHIE and BUGGY (Brown & Burton, 1975, 1978).

Such an analysis can suggest another solution to some teaching problems: redesign the domain you are teaching to match the students' prior model. This is not feasible in many domains, but quite possible in applications involving computer software: new programming languages, text editors, message systems, etc. For example, the text editor used here could be redesigned to check when in append mode whether a line typed in is likely to be a command, and question the user as to his or her intent -- in essence design the editor to act like the secretary model that the user is imposing on it.

Finally, an attempt has been made within this theory to push the model of the learning process to the point that it can be implemented on a computer, and I have begun work on such an implementation. Such a model if completed, provides not only an analysis of the learning behavior of the student, but a model to a teacher, also implemented on a computer, from which perspective it can much better teach.

I believe these applications suggest the potentially significant contribution of cognitive psychology to teaching. But there is much work ahead.



## Bibliography


Bobrow, D. G., & Norman, D. A. Some principles of memory schemata. In D. G. Bobrow & A. M. Collins (Eds.), Representation and understanding: Studies in Cognitive Science. New York: Academic Press, 1975.

Bobrow, D. G., & Winograd, T. An overview of KRL: A knowledge representation language. Cognitive Science, 1977, 1, 3-46.

Brachman, R. J. A structural paradigm for representing knowledge (Tech. Rep. 3605). Boston: Bolt, Beranek & Newman, 1978.

Brown, J. S., & Burton, R. R. Multiple representation of knowledge for tutorial reasoning. In D. G. Bobrow & A. M. Collins (Eds.), Representation and understanding: Studies in cognitive science. New York: Academic Press, 1975.

Brown, J. S. & Burton, R. R. Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 1978, 2, 155-192.

Charniak, E. A framed PAINTING: The representation of a common sense knowledge fragment. Cognitive Science, 1977, 1, 355-394.

Gentner, D. Evidence for the psychological reality of semantic components: The verbs of possession. In D. A. Norman, D. E. Rumelhart, & The LNR Research Group, Explorations in cognition. San Francisco: Freeman, 1975.

Grice, P. H. Logic and conversation. In P. Cole & J. L. Morgan (Eds.), Syntax and semantics (Vol. 3). New York: Academic Press, 1975.

Minsky, M. A framework for representing knowledge. In P. H. Winston (Ed.), The psychology of computer vision. New York: McGraw-Hill, 1975.

Newell, A., & Simon, H. Human problem solving. Englewood Cliffs, N.J.: Prentice-Hall, 1972.

Norman, D. A., Rumelhart, D. E., & the LNR Research Group. Explorations in cognition. San Francisco: Freeman, 1975

Palmer, S. E. Fundamental aspects of cognitive representation: Theory and metatheory. In E. Rosch (Ed.), ?. Hillsdale, N. J.: Erlbaum Associates, in press.

Moore, J., & Newell, A. How can MERLIN understand? In L. W. Gregg (Ed.), Knowledge and cognition. Potomac, Md.: Erlbaum Associates, 1974.

Rumelhart, D. E., & Norman, D. A. Accretion, tuning and restructuring: Three modes of learning. In R. Klatsky and J. W. Cotton (Eds.), Semantic factors in cognition. Hillsdale, N. J.: Erlbaum Associates, in press.

Rumelhart, D. E., & Ortony, A. *The representation of knowledge in memory*. In R. C. Anderson, R. J. Spiro, & W. E. Montague (Eds.), Schooling and the acquisition of knowledge. Hillsdale, N. J.: Erlbaum Associates, 1977.

Schank, R. C. Identification of conceptualizations underlying natural language. In R. C. Schank & K. M. Colby (Eds.), Computer models of thought and language. San Francisco: Freeman, 1973.

Schank, R. C., & Abelson, R. P. Scripts, plans, and knowledge. Advance Papers of the Fourth International Joint Conference on Artificial Intelligence. Tbilisi, Georgia, USSR, 1975, pp. 151-157.

Woods, W. A. What's in a link: Foundations for semantic networks. In D. G. Bobrow & A. Collins (Eds.), Representation and understanding: Studies in cognitive science. N. Y.: Academic Press, 1975.



## Footnotes

1. The symbol <CR> stands for the "RETURN" or "CARRIAGE RETURN" key. The symbol [n] is used here to refer to some integer, representing a line number.

2. I have not shown the minimality or necessary nature of all of these structures, and in some senses the structures certainly are not minimal. One appropriate criterion for testing minimality might be called functional completeness: given a particular editing task, a correct sequence (there may be more than one) of editing commands are executed. It is possible to satisfy this criterion with a direct mapping of particular errors in the text to specific character strings to type. However, from the instruction manual it is clear that this is not what the teachers intended to teach.

3. The same phenomenon often happens to me when reading a book about a mathematics area unknown to me: The first few pages, while superficially very straightforward, often cause more problems than later.

4. In effect, it bears a much closer resemblance to the concept of "accretion," used in Rumelhart & Norman, 1977. However, the descriptions are meant to be at different levels, and again I believe that the extent of use here is much more limited. Remember, the point here is the automatic assumption of the prior nature of new knowledge.

## Appendix A

### On Building Models of Learning and Teaching

One can develop a cognitive learning theory for many possible reasons or goals, not all of which may be realizable. Thus it is perhaps useful at this point to consider the role of such a cognitive learning theory in the context of earlier psychological and educational theories of learning.

Without going into such earlier theories in detail it is certain that they have had the benefit of much trial-and-error and experience about how best to go about learning a knowledge domain. Educational psychologists in particular have developed effective ways of teaching various areas and have many years of actual education to back them up. Prescriptions of the character, "It is important that students work out for themselves sample problems in the domain.", or, "The material to be taught should be clearly organized." have been well established.

For this reason it is highly unlikely that a correct cognitive theory of learning will contradict many of these general prescriptions. Thus it is perhaps an unreasonable goal that a cognitive theory will revolutionize learning and teaching methods in this sense.

What function, then, does a cognitive theory serve? The prescriptions such as those above empirically derived from education in a sense provide a "surface structure" for teaching into which any theory of the underlying structure must fit, much as any description of a whole iceberg must fit what shows above the water. Thus a cognitive theory makes a set of assumptions about learning at some "deeper" level -- where deeper is closer to basic information processing or neural mechanisms -- and derives implications from these assumptions to build an internal structure of a learning theory as well as predicting the same surface prescriptions derived from education.

There are at least two major objections to such a predictive process. First, if the cognitive theory makes the same "visible" predictions as current educational theories, then nothing of practical use has been added to how to go about learning or teaching. One can point out that there would then be a theoretical reason for believing the educational theories, and that one need experiment no further. However, even this support is dependent upon the correctness of the deeper level assumptions.

Second, to the extent that the assumptions on which the cognitive theory is based are arbitrary and can be varied, the theory provides no explanation at all, even on a purely logical level. A general form of this problem of course occurs in all scientific model building. The arbitrariness of the assumptions is usually restricted by comparison to related domains, and by considerations of "parsimony." In formulating a cognitive theory of learning, neither of these restrictions are as





effective: Learning complex domains involves considering complex structures, with no strict relation to nearby domains, nor incentive to obey parsimony.

The objection that learning requires complex structures is contrasted by the vagueness of the prescriptions of educational psychology, and the lack of precise specification of when they should be used. A cognitive theory of learning and teaching should formally and strictly define each of these prescriptions and when they apply. To extend the iceberg analogy, one is depicting the bumps of the icebergs in detail rather than the general shape. Second, an underlying cognitive theory may tie together many of these prescriptions, and perhaps point out a few which are inconsistent with the rest and maybe less correct.

Also, in specifying when particular learning and teaching methods should apply, one ideally is defining an algorithm by which a particular knowledge domain can be most efficiently (or maximally in some other way) taught. This certainly is a laudable goal for devising a cognitive theory; however, I suspect that in many senses it is not achievable. An attempt will be made to prove this in some cases later.

The second objection is perhaps more serious. Given a surface structure of visible behavior and empirical methods, one can proceed in building an underlying structure by working down from the surface and building only structure which is consistent with that surface. One can also intelligently choose a set of assumptions, and attempt to derive a surface structure from them. An interactive approach is also possible. However, in the case of teaching and learning, neither method may be feasible: In building down from a complicated surface, the internal structures are likely to be also complicated, and this in turn usually allows a wide but arbitrary choice of possible internal structures which fit. There is a close mathematical analogy to this: given a particular domain and range there are relatively few "simple" mappings between them. However, if one is allowed to choose a complicated mapping, the choice usually becomes immense. In the case of learning, a complicated surface structure forces a complicated "mapping."

A similar argument counters the "axiom-first" approach: Since the surface structure is so complicated, it is extremely unlikely that a simple set of assumptions will produce such a surface. Finally, one can begin with a simple set of assumptions and work interactively from the surface and axioms. However, the restrictions one must make in order to make the surface derivable from the assumptions are often at least as arbitrary and sometimes more complicated that the strict "surface-down" approach.

What then is a reasonable goal for a cognitive model of learning and teaching? Several feasible approaches will be described below, and although they are somewhat general, I'm sure that they do not cover the possibilities. First, one major problem mentioned above is the arbitrariness of the choices which must be made in constructing an internal structure to match a complicated surface. One alternative to this, when the choice is arbitrary or nearly so, is not to make the choice.

Instead the theory generated by each choice is considered as a point in a theory space. If one defines a "second-level theory" as a theory space generated by one or more choice points, then one can consider the structure of this type of theory instead. It may be true that the variation caused by the choice point has no effect on the surface structure, i.e., the visible behavior. In essence there is no basis for the choice and it need not be made. Often this is not clear unless considered within the structure of a second-level theory. (Note that these last two statements are always with the qualification, "given a particular quality of data collected on behavior." A data-collection of a more detailed nature may force a decision at one or more of the choice points.)

As mentioned earlier, ideally one would want an algorithm for teaching such that, given any knowledge domain and student, the algorithm will generate the most effective teaching sequence that is possible. There are many necessary but quite difficult subgoals within such an algorithm. First it is necessary to completely define the structure of the knowledge domain to be taught. While this is usually feasible in an artificial domain such as mathematics, the task may be theoretically impossible in some domains with many references to the real world.

Secondly, even given a case where a complete representation of the domain is possible, there are usually many such representations which are logically equivalent. However, they are not equivalent at all within the total context of particular students with different current knowledge structures and possibly different learning processes. Therefore, the interaction *between student and domain* must be considered. This in turn *ideally requires complete* knowledge of the student's current structure. Thus, such an algorithm must include a testing procedure for obtaining such knowledge.

The algorithm just described is an extreme case, one which guarantees learning of a domain maximally. Nevertheless, it is useful to describe the characteristics of such an algorithm, in order to establish one endpoint of a dimension along which cognitive theories of learning can vary. Also, given that such an algorithm is very likely not attainable, it is useful to establish out in front that such a goal is not reasonable, for I suspect that such a goal has still not been clearly rejected by some learning theorists.

Surely, however, people would be satisfied with a less lofty goal. Perhaps it is possible to devise a heuristic which tends to maximize the learning of a knowledge domain on some statistical measure. The goal can be weakened even further to just "approaching maximization." One must realize, however, that this is approaching the neighborhood of the prescriptions from educational psychology, heuristics which often help the process of teaching and learning. The difference between these and the best heuristics is only a matter of degree.

Therefore, in terms of practical results, the only reasonable goal for a cognitive psychologist in learning is to develop known teaching heuristics which are closer to the best possible ones. Most



cognitive theorists, however, would argue that their major goal is at a different level -- to build a learning theory which is consistent with what we know about the internal structure of the human information processing mechanism. However, I have tried to argue above that a complete accomplishment of this goal is chancy at best: Such a theory is likely not to make contact with the surface structure (visible behavior) without the addition of many arbitrary assumptions. Nevertheless, this thesis is an attempt at the the second goal.

## Appendix B

Below is a complete listing of the instruction manual used.

### Simple ed: 'end' version

--------------------------------------------------------------

(press the "space" bar at the bottom of the keyboard
to get to the next page.)

"GENERAL INSTRUCTIONS"
We are studying how people understand instruction manuals. We are going to present an instruction manual to you, one sentence at a time.

As you read the instructions, please tell us, aloud,

What each sentence means
What you now understand
What you think will happen next

To summarize:
After each sentence, tell us

Its meaning
Your understanding
What happens next

When you are asked to do something, you should do it on the terminal to your left. This terminal is used for the instruction manual.

ok. Let us begin.

--------------------------------------------------------------

We are going to teach you how to edit a text.

The text is stored in a computer.



There are a number of mistakes in the text.

We would like you to correct them.

THE p COMMAND

How to print text.

Type "3p" on the terminal to your left.

Type the key marked RETURN.

Every command must end with RETURN.

Notice the screen.

The 3rd line of the text has been printed.

Now try printing the fifth line.

You should have typed "5p (RETURN)"

This would have displayed the 5th line.

THE QUESTION MARK (?)

Type "20p"

Notice that you go a "?"

This is because there are less than 20 lines in this particular text.

The ? means that the computer did not understand.

(Did you remember to type a RETURN after you typed the 20p ?)

A SIMPLE SET OF EXERCISES

Print the first line.

Print the third line.

Print lines 1 through 5

PRINTING A GROUP OF LINES

There is an easy way to see a group of lines.

Type "1,5p"

This shows you the first five lines.

PRINTING THE LAST LINE: $

Suppose you don't know what the number of the last line is.

The symbol, $ stands for the last line.

type "$"

This gives you the last line of the text.

Please display the entire text.

PRINTING THE ENTIRE TEXT

To see the entire text, type: "1,$p"

This means to print from lines 1 through line $ (the last line).

THE d COMMAND

To delete a line use the d command.

" d " stands for delete.

Notice that the 3rd and 4th lines of the text are the same.

To delete line 4, simply type "4d (followed by RETURN)"

Try it.

Now print the entire text again.

Note that there is no duplication of lines.

You tell the d command what lines to delete in the same way that you tell the p command what lines to print.

Thus, to delete the last line, you would type "$d". (But please don't do it.)

To delete a group of lines, such as lines 25 through 35, you would type "25,35d"

THE a COMMAND

If you wanted to add some material to the text, you could use the a command: 'a' stands for "append."

For example, suppose that we wished to add a paragraph to the text right after line 7.

To do this you need to know several things.

So, don't do anything until we tell you to.



Once you give the editor the command "7a" everything you type will be added to the text after line 7.

The only way to stop appending is to type a line that contains only the word "end".

The word "end" is used to tell the editor that you have finished appending.

A SAMPLE TEXT

Try adding the following sentence to the text after line 4.

It is not a simple activity.

Now print the entire text again.

CORRECTING TYPING ERRORS: BACKSPACE

If you make a typing error, you can correct it by backspacing over the error.

To backspace, hold down the key on the left side of the keyboard marked CTRL (for control) while you type ' h '.

Notice the left arrow on the top of the 'h' key.

(From now on, we will simply speak of backspace as control-h.)

If you decide that the entire line that you are typing is wrong, you can correct the entire line by typing the symbol "".

Everything typed on the same line to the left of the is ignored -- as if it weren't typed.

ADDING TO THE END OF THE TEXT

Now, add some material to the end of the text.

Please add the following:

Accretion is a never-ending process, for complex materials are rich in the amount of knowledge they contain. Restructuring is infrequent, by essential for true understanding. And tuning requires hours of practice.

Print out the entire text.

If you find typing errors, you should correct them by deleting the lines and appending the correct line.

Please correct all the typing errors.

TASK 3: APPENDING TO THE MIDDLE OF THE TEXT

Now we would like you to do another task.

We want you to add four words to the text, right after line 10.

Put each word on a separate line.

NOTE: AFTER YOU HAVE TYPED THE FIRST TWO WORDS, PRINT OUT THE ENTIRE TEXT SO WE CAN SEE IT.

THEN, FINISH ALL FOUR LINES AND AGAIN, PRINT OUT ALL THE TEXT.

Here are the four words (remember, put each on a separate line, starting after line 10):

schemata
porpoise
Socrates
instantiation

That's all.

Thank you for working with us.

## Appendix C

### On Obtaining and Analyzing Protocols

It has been said that protocols are an easy form of data to obtain, but a very difficult form to perform any significant analysis on. I agree with only half of this statement -- I believe that there are almost as many problems in obtaining good protocols as in analyzing them. Given the difficulty of this data form, it is worthwhile to consider briefly whether it is necessary here. I am trying to analyze and model the learning process on as close to an instant to instant basis as possible -- to describe each set of intermediate internal structures which occur within the student as he or she reads through an instruction manual. Therefore, I need as much data about the thinking processes at each moment as possible; and the protocol is by far the most efficient, and perhaps the only method, of getting this. In fact, as will result from the analysis, the protocol is much to sparse of a data form for what is really needed.

Obtaining learning protocols, perhaps more than getting protocols in other situations (e.g. problem solving, Newell & Simon, 1973), is subject to a form of "Heisenberg Principle of psychology": the process of getting them tends to modify the learning process. There are several manifestations of this: Asking subjects about a particular concept tends to activate that concept, and perhaps cause it to play a more significant role in the learning process than it would have. Questioning a subject about a the details of a definition may cause a reorganization of their knowledge about that concept, even if the experimenter supplies no information. Asking a subject how or why about a particular learning point in the process, alerts the subject to the significance of that point, and may cause the inference that something was learned wrong, even if the experimenter didn't intend this. (There are many other examples.)

However, the experimenter cannot just let the subject talk aloud: Most subjects will not say nearly enough about their internal processes, perhaps more in learning where many of the processes are automatic, than in problem solving where the subject may be aware of many of the steps gone through. And even practiced subjects will not talk about what is needed for the analysis, describing mostly their conclusions rather than how they arrived at them. If they do, their learning processes have often been modified as a result.

Perhaps the key problem is that much of the process of learning is automatic or semi-automatic, of which the learner "sees" only the results. In addition, many of these small learning "results" are seen as obvious by the learner to both themselves and the teacher: By conversational postulates these are rarely stated. Yet it is these small results and the process underlying them which are the key to the modeling the learning process. (An interesting problem which occurs because of this is, when I question them about a point which is "obvious," they

often read much more into the question than is intended. In these cases, they either cannot come up with an answer, or begin an extraneous or incorrect inference process on what they thought was implied.)

These constraints from either side, getting as much data about the learning process occurring as possible, and yet not affecting that process, cannot both be satisfied. The only solution is some sort of compromise: Presenting a manual one sentence at a time and questioning them extensively, may affect the learning process some, but hopefully not to a great extent, and not the essential character. Considerable effort was spent in not transmitting information in questions asked of the student:

(1) Using the same words to refer to concepts as the student used. Synonyms to the questioner often have different connotations to the learner.
(2) Asking questions about parts that I had no interest in, in order not to signal the significance of parts that I was.
(3) Asking questions in the same form whether the student was correct or incorrect.

There are occasional slips. A couple are noted in the analysis in Example 1.

<u>Analyzing Protocols</u>

The general approach to protocol analysis taken here is perhaps more "naturalistic" than earlier analysis (e.g., Newell & Simon, 1973). No attempt is made to try to classify each of the subject's statements into a set of categories, or to count the number of event or concept occurrences of a particular type. The view taken here might be described as though the subject's data is the surface of an iceberg visible above water, and the goal is to infer the underwater structure. The visible surface of the iceberg is irregularly-shaped and must be taken as a whole. Thus, the approach here is to interpret each statement of the learner sequentially, as the next piece of evidence of a continuing underlying process.

There is a possible introduction of bias by interpreting protocol in this naturalistic fashion. However, I believe that this bias tends to be balanced by the biases introduced by dissecting protocols into a set of categories: The protocol was initially generated in a linear order, such that consecutive sentences tend to be dependent upon each other.

The analysis here concentrates almost solely on the protocol of one subject, and one could object that any conclusions gotten could hardly be extendible. However, the goal of this thesis is <u>not</u> to extend the conclusions, but to suggest a theory and set of methodologies for studying the process of learning. The theory itself is not being tested, what is being tested is whether a methodology can be developed within such a theoretical framework to study the process of complex learning.

Navy

1 Dr. Ed Aiken
Navy Personnel R&D Center
San Diego, CA 92152

1 Dr. Jack R. Borsting
Provost & Academic Dean
U.S. Naval Postgraduate School
Monterey, CA 93940

1 Dr. Robert Breaux
Code N-71
NAVTRAEQUIPCEN
Orlando, FL 32813

1 Dr. Richard Elster
Department of Administrative Sciences
Naval Postgraduate School
Monterey, CA 93940

1 DR. PAT FEDERICO
NAVY PERSONNEL R&D CENTER
SAN DIEGO, CA 92152

1 CDR John Ferguson, MSC, USN
Naval Medical R&D Command (Code 44)
National Naval Medical Center
Bethesda, MD 20014

1 Dr. John Ford
Navy Personnel R&D Center
San Diego, CA 92152

1 Dr. Steve Harris
Code L522
NAMRL
Pensacola FL 32508

1 CDR Robert S. Kennedy
Naval Aerospace Medical and
Research Lab
Box 29407
New Orleans, LA 70189

1 Dr. Norman J. Kerr
Chief of Naval Technical Training
Naval Air Station Memphis (75)
Millington, TN 38054

1 Dr. William L. Maloy
Principal Civilian Advisor for
Education and Training
Naval Training Command, Code 00A
Pensacola, FL 32508

1 CAPT Richard L. Martin
USS Francis Marion (LPA-Z49)
FPO New York, NY 09501

2 Dr. James McGrath
Navy Personnel R&D Center
Code 306
San Diego, CA 92152

1 DR. WILLIAM MONTAGUE
LRDC
UNIVERSITY OF PITTSBURGH
3939 O'HARA STREET
PITTSBURGH, PA 15213

1 Naval Medical R&D Command
Code 44
National Naval Medical Center
Bethesda, MD 20014

1 CAPT Paul Nelson, USN
Chief, Medical Service Corps
Code 7
Bureau of Medicine & Surgery
U. S. Department of the Navy
Washington, DC 20372

1 Library
Navy Personnel R&D Center
San Diego, CA 92152

6 Commanding Officer
Naval Research Laboratory
Code 2627
Washington, DC 20390

1 JOHN OLSEN
CHIEF OF NAVAL EDUCATION &
TRAINING SUPPORT
PENSACOLA, FL 32509

1 Psychologist
ONR Branch Office
495 Summer Street
Boston, MA 02210

1 Psychologist
ONR Branch Office
536 S. Clark Street
Chicago, IL 60605

1 Office of Naval Research
Code 200
Arlington, VA 22217

1 Office of Naval Research
Code 437
800 N. Quincy SStreet
Arlington, VA 22217

5 Personnel & Training Research Programs
(Code 458)
Office of Naval Research
Arlington, VA 22217

1 Psychologist
OFFICE OF NAVAL RESEARCH BRANCH
223 OLD MARYLEBONE ROAD
LONDON, NW, 15TH ENGLAND

1 Psychologist
ONR Branch Office
1030 East Green Street
Pasadena, CA 91101

1 Scientific Director
Office of Naval Research
Scientific Liaison Group/Tokyo
American Embassy
APO San Francisco, CA 96503

1 Head, Research, Development, and Studies
(OP102X)
Office of the Chief of Naval Operations
Washington, DC 20370

1 Scientific Advisor to the Chief of
Naval Personnel (Pers-Or)
Naval Bureau of Personnel
Room 4410, Arlington Annex
Washington, DC 20370

1 DR. RICHARD A. POLLAK
ACADEMIC COMPUTING CENTER
U.S. NAVAL ACADEMY
ANNAPOLIS, MD 21402

1 Mr. Arnold Rubenstein
Naval Personnel Support Technology
Naval Material Command (08T244)
Room 1044, Crystal Plaza #5
2221 Jefferson Davis Highway
Arlington, VA 20360

1 Dr. Worth Scanland
Chief of Naval Education and Training
Code N-5
NAS, Pensacola, FL 32508

1 A. A. SJOHOLM
TECH. SUPPORT, CODE 201
NAVY PERSONNEL R& D CENTER
SAN DIEGO, CA 92152

1 Mr. Robert Smith
Office of Chief of Naval Operations
OP-987E
Washington, DC 20350

1 Dr. Alfred F. Smode
Training Analysis & Evaluation Group
(TAEG)
Dept. of the Navy
Orlando, FL 32813

1 CDR Charles J. Theisen, JR. MSC, USN
Head Human Factors Engineering Div.
Naval Air Development Center
Warminster, PA 18974

1 W. Gary Thomson
Naval Ocean Systems Center
Code 7132
San Diego, CA 92152

Army

1 Technical Director
U. S. Army Research Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

1 HQ USAREUE & 7th Army
ODCSOPS
USAAREUE Director of GED
APO New York 09403

1 DR. RALPH DUSEK
U.S. ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 Col. Frank Hart, Director
Training Development Institute
ATTNG-TDI
Ft. Eustis, VA 23604

1 Dr. Ed Johnson
Army Research Institute
5001 Eisenhower Blvd.
Alexandria, VA 22333

1 Dr. Michael Kaplan
U.S. ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 Dr. Harold F. O'Neil, Jr.
ATTN: PERI-OK
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 LTCOL Michael T. Plummer
Organizational Effectiveness Division
Office of the Deputy Chief of Staff
for Personnel
Department of the Army
Washington, DC 20301

1 Dr. Joseph Ward
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Air Force

1 Air University Library
AUL/LSE 76/443
Maxwell AFB, AL 36112

1 DR. T. E. COTTERMAN
AFHRL/ASR
WRIGHT PATTERSON AFB
OHIO 45433

1 DR. G. A. ECKSTRAND
AFHRL/AS
WRIGHT-PATTERSON AFB, OH 45433

1 CDR. MERCER
CNET LIAISON OFFICER
AFHRL/FLYING TRAINING DIV.
WILLIAMS AFB, AZ 85224

1 Dr. Donald E. Meyer
U.S. Air Force
ATC/XPTD
Randolph AFB, TX 78148

1 Dr. Ross L. Morgan (AFHRL/ASR)
Wright -Patterson AFB
Ohio 45433

1 Dr. Marty Rockway (AFHRL/TT)
Lowry AFB
Colorado 80230

1 Jack A. Thorpe, Capt, USAF
Program Manager
Life Sciences Directorate
AFOSR
Bolling AFB, DC 20332

Marines

1 MCDEC
Quantico Marine Corps Base
Quantico, VA 22134

1 DR. A.L. SLAFKOSKY
SCIENTIFIC ADVISOR (CODE RD-1)
HQ, U.S. MARINE CORPS
WASHINGTON, DC 20380

CoastGuard

1 MR. JOSEPH J. COWAN, CHIEF
PSYCHOLOGICAL RESEARCH (G-P-1/62)
U.S. COAST GUARD HQ
WASHINGTON, DC 20590

Other DoD

1 Dr. Stephen Andriole
ADVANCED RESEARCH PROJECTS AGENCY
1400 WILSON BLVD.
ARLINGTON, VA 22209

12 Defense Documentation Center
Cameron Station, Bldg. 5
Alexandria, VA 22314
Attn: TC

1 Dr. Dexter Fletcher
ADVANCED RESEARCH PROJECTS AGENCY
1400 WILSON BLVD.
ARLINGTON, VA 22209

1 Military Assistant for Training and
Personnel Technology
Office of the Under Secretary of Defense
for Research & Engineering
Room 3D129, The Pentagon
Washington, DC 20301

Civil Govt

1 Dr. Susan Chipman
Basic Skills Program
National Institute of Education
1200 19th Street NW
Washington, DC 20208

11 Mr. James M. Ferstl
Bureau of Training
U.S. Civil Service Commission
Washington, D.C. 20415

1 Dr. Joseph I. Lipson
Division of Science Education
Room W-638
National Science Foundation
Washington, DC 20550

1 Dr. John Mays
National Institute of Education
1200 19th Street NW
Washington, DC 20208

1 William J. McLaurin
Rm. 301, Internal Revenue Servic
2221 Jefferson Davis Highway
Arlington, VA 22202

1 Dr. Arthur Melmed
National Intitute of Education
1200 19th Street NW
Washington, DC 20208

1 Dr. Andrew R. Molnar
Science Education Dev.
and Research
National Science Foundation
Washington, DC 20550

1 Dr. H. Wallace Sinaiko
Program Director
Manpower Research and Advisory Services
Smithsonian Institution
801 North Pitt Street
Alexandria, VA 22314

1 Dr. Thomas G. Sticht
Basic Skills Program
National Institute of Education
1200 19th Street NW
Washington, DC 20208

1 Dr. Frank Withrow
U. S. Office of Education
400 6th Street SW
Washington, DC 20202

1 Dr. Joseph L. Young, Director
Memory & Cognitive Processes
National Science Foundation
Washington, DC 20550

Non Govt

1 Dr. Earl A. Alluisi
HQ, AFHRL (AFSC)
Brooks AFB, TX 78235

1 Dr. John R. Anderson
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

1 DR. MICHAEL ATWOOD
SCIENCE APPLICATIONS INSTITUTE
40 DENVER TECH. CENTER WEST
7935 E. PRENTICE AVENUE
ENGLEWOOD, CO 80110

1 1 psychological research unit
Dept. of Defense (Army Office)
Campbell Park Offices
Canberra ACT 2600, Australia

1 Dr. R. A. Avner
University of Illinois
Computer-Based Educational Research Lab
Urbana, IL 61801

1 Dr. Alan Baddeley
Medical Research Council
Applied Psychology Unit
15 Chaucer Road
Cambridge CB2 2EF
ENGLAND

1 Ms. Carole A. Bagley
Minnesota Educational Computing
Consortium
2520 Broadway Drive
St. Paul, MN 55113

1 Mr Avron Barr
Department of Computer Science
Stanford University
Stanford, CA 94305

1 Dr. John Bergan
School of Education
University of Arizona
Tuscon AZ 85721

1 Dr. Nicholas A. Bond
Dept. of Psychology
Sacramento State College
600 Jay Street
Sacramento, CA 95819

1 Dr. Lyle Bourne
Department of Psychology
University of Colorado
Boulder, CO 80302

1 Dr. Kenneth Bowles
Institute for Information Sciences
University of California at San Diego
La Jolla, CA 92037

1 Dr. John S. Brown
XEROX Palo Alto Research Center
3333 Coyote Road
Palo Alto, CA 94304

1 Dr. Bruce Buchanan
Department of Computer Science
Stanford University
Stanford, CA 94305

1 DR. C. VICTOR BUNDERSON
WICAT INC.
UNIVERSITY PLAZA, SUITE 10
1160 SO. STATE ST.
OREM, UT 84057

1 Dr. Anthony Cancelli
School of Education
University of Arizona
Tuscon, AZ 85721

1 Dr. John B. Carroll
Psychometric Lab
Univ. of No. Carolina
Davie Hall 013A
Chapel Hill, NC 27514

Non Govt

1 Charles Myers Library
Livingstone House
Livingstone Road
Stratford
London E15 2LJ
ENGLAND

1 Dr. William Chase
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

1 Dr. Micheline Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

1 Dr. John Chiorini
Litton-Mellonics
Box 1286
Springfield, VA 22151

1 Dr. William Clancey
Department of Computer Science
Stanford University
Stanford, CA 94305

1 Dr. Allan M. Collins
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, Ma 02138

1 Dr. Meredith Crawford
Department of Engineering Administration
George Washington University
Suite 805
2101 L Street N. W.
Washington, DC 20037

1 Dr. Hubert Dreyfus
Department of Philosophy
University of California
Berkely, CA 94720

1 ERIC Facility-Acquisitions
4833 Rugby Avenue
Bethesda, MD 20014

1 Dr. A. J. Eschenbrenner
Dept. E422, Bldg. 101
McDonnell Douglas Astronautics Co.
P.O.Box 516
St. Louis, MO 63166

1 MAJOR I. N. EVONIC
CANADIAN FORCES PERS. APPLIED RESEARCH
1107 AVENUE ROAD
TORONTO, ONTARIO, CANADA

1 Dr. Ed Feigenbaum
Department of Computer Science
Stanford University
Stanford, CA 94305

1 Dr. Richard L. Ferguson
The American College Testing Program
P.O. Box 168
Iowa City, IA 52240

1 Mr. Wallace Feurzeig
Bolt Beranek & Newman, Inc.
50 Moulton St.
Cambridge, MA 02138

1 Dr. Victor Fields
Dept. of Psychology
Montgomery College
Rockville, MD 20850

1 Dr. Edwin A. Fleishman
Advanced Research Resources Organ.
Suite 900
4330 East West Highway
Washington, DC 20014

1 Dr. John R. Frederiksen
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

1 DR. ROBERT GLASER
LRDC
UNIVERSITY OF PITTSBURGH
3939 O'HARA STREET
PITTSBURGH, PA 15213

1 Dr. Ira Goldstein
XEROX Palo Alto Research Center
3333 Coyote Road
Palo Alto, CA 94304

1 DR. JAMES G. GREENO
LRDC
UNIVERSITY OF PITTSBURGH
3939 O'HARA STREET
PITTSBURGH, PA 15213

1 Dr. Ron Hambleton
School of Education
University of Massechusetts
Amherst, MA 01002

1 Dr. Barbara Hayes-Roth
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406

1 Dr. Frederick Hayes-Roth
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406

1 Dr. Dustin H. Heuston
Wicat, Inc.
Box 986
Orem, UT 84057

1 Dr. James R. Hoffman
Department of Psychology
University of Delaware
Newark, DE 19711

1 Library
HumRRO/Western Division
27857 Berwick Drive
Carmel, CA 93921

1 Dr. Earl Hunt
Dept. of Psychology
University of Washington
Seattle, WA 98105

1 Mr. Gary Irving
Data Sciences Division
Technology Services Corporation
2811 Wilshire Blvd.
Santa Monica CA 90403

1 DR. LAWRENCE B. JOHNSON
LAWRENCE JOHNSON & ASSOC., INC.
SUITE 502
2001 S STREET NW
WASHINGTON, DC 20009

1 Dr. Wilson A. Judd
McDonnell-Douglas
Astronautics Co. East
Lowry AFB
Denver, CO 80230

1 Dr. Steven W. Keele
Dept. of Psychology
University of Oregon
Eugene, OR 97403

1 Dr. Walter Kintsch
Department of Psychology
University of Colorado
Boulder, CO 80302

1 Dr. David Kieras
Department of Psychology
University of Arizona
Tuscon, AZ 85721

1 Mr. Marlin Kroger
1117 Via Goleta
Palos Verdes Estates, CA 90274

1 LCOL. C.R.J. LAFLEUR
PERSONNEL APPLIED RESEARCH
NATIONAL DEFENSE HQS
101 COLONEL BY DRIVE
OTTAWA, CANADA K1A 0K2

1 Dr. Jill Larkin
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

1 Dr. Alan Lesgold
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

1 Dr. Robert A. Levit
Manager, Behavioral Sciences
The BDM Corporation
7915 Jones Branch Drive
McClean, VA 22101

1 Dr. Robert R. Mackie
Human Factors Research, Inc.
6780 Cortona Drive
Santa Barbara Research Pk.
Goleta, CA 93017

1 Dr. Mark Miller
Systems and Information Sciences Lab
Central Research Laboratories
TEXAS INSTRUMENTS, INC.
Mail Station 5
Post Office Box 5936
Dallas, TX 75222

1 Dr. Richard B. Millward
Dept. of Psychology
Hunter Lab.
Brown University
Providence, RI 82912

1 Dr. Stuart Milner
Department of Education
George Mason University
4400 Fairfax Drive
Fairfax, VA 22030

1 Dr. Allen Munro
Univ. of So. California
Behavioral Technology Labs
3717 South Hope Street
Los Angeles, CA 90007

1 Dr. Seymour A. Papert
Massachusetts Institute of Technology
Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139

1 Dr. James A. Paulson
Portland State University
P.O. Box 751
Portland, OR 97207

1 MR. LUIGI PETRULLO
2431 N. EDGEWOOD STREET
ARLINGTON, VA 22207

1 Dr. Barbara Pflanz
Department of German
University of Redlands
Redlands, CA 92373

1 DR. PETER POLSON
DEPT. OF PSYCHOLOGY
UNIVERSITY OF COLORADO
BOULDER, CO 80302

1 DR. DIANE M. RAMSEY-KLEE
R-K RESEARCH & SYSTEM DESIGN
3947 RIDGEMONT DRIVE
MALIBU, CA 90265

1 Dr. Peter B. Read
Social Science Research Council
605 Third Avenue
New York, NY 10016

1 Dr. Fred Reif
SESAME
c/o Physics Department
University of California
Berkely, CA 94720

1 Dr. Andrew M. Rose
American Institutes for Research
1055 Thomas Jefferson St. NW
Washington, DC 20007

1 Dr. Ernst Z. Rothkopf
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

1 Dr. David Rumelhart
Center for Human Information Processing
Univ. of California, San Diego
La Jolla, CA 92093

1 DR. WALTER SCHNEIDER
DEPT. OF PSYCHOLOGY
UNIVERSITY OF ILLINOIS
CHAMPAIGN, IL 61820

1 Dr. Allen Schoenfeld
Department of Mathematics
Hamilton College
Clinton, NY 13323

1 DR. ROBERT J. SEIDEL
INSTRUCTIONAL TECHNOLOGY GROUP
HUMRRO
300 N. WASHINGTON ST.
ALEXANDRIA, VA 22314

1 Dr. Richard Snow
School of Education
Stanford University
Stanford, CA 94305

1 Dr. Robert Sternberg
Dept. of Psychology
Yale University
Box 11A, Yale Station
New Haven, CT 06520

1 DR. ALBERT STEVENS
BOLT BERANEK & NEWMAN, INC.
50 MOULTON STREET
CAMBRIDGE, MA 02138

1 Mr. William Stobie
McDonnell-Douglas
Astronautics Co.
P. O. Box 30204
Chico, CA 95926

1 DR. PATRICK SUPPES
INSTITUTE FOR MATHEMATICAL STUDIES IN
THE SOCIAL SCIENCES
STANFORD UNIVERSITY
STANFORD, CA 94305

1 Dr. John Thomas
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

1 DR. PERRY THORNDYKE
THE RAND CORPORATION
1700 MAIN STREET
SANTA MONICA, CA 90406

1 Dr. Walt W. Tornow
Control Data Corporation
Corporate Personnel Research
P.O. Box 0 - HQN060
Minneapolis, MN 55440

1 Dr. Douglas Towne
Univ. of So. California
Behavioral Technology Labs
3717 South Hope Street
Los Angeles, CA 90007

1 Dr. Benton J. Underwood
Dept. of Psychology
Northwestern University
Evanston, IL 60201

1 Dr. David J. Weiss
N660 Elliott Hall
University of Minnesota
75 E. River Road
Minneapolis, MN 55455

1 Dr. Karl Zinn
Center for research on Learning
and Teaching
University of Michigan
Ann Arbor, MI 48104